TRABAJO DE FIN DE GRADO

Aplicación para entornos empresariales.

Autor: Pablo Camprubí García
Tutor: Israel González Carrasco

Grado: Doble Grado en Administración y dirección de empresas e Ingeniería Informática

Colmenarejo, julio de 2019

Agradecimientos.

Me gustaría agradecer a la Universidad Carlos III de Madrid la formación que me ha dado durante estos últimos años, no solo en términos académicos, sino también en valores.

Estoy orgulloso de poder decir que esta siempre será mi casa, donde aprendí a trabajar de verdad, a esforzarme, a caer y a levantarme, así como dejarme levantar por otros y tener la oportunidad de ayudar a muchos otros tantos.

Gracias también a todos los profesores que lo han hecho posible, compañeros que me acompañaron durante este camino y sobre todo mis padres y hermanos, quienes ante viento y marea me han apoyado hasta el final para terminar la universidad.

Por último, me gustaría poner nombres y apellidos a las personas mas especiales sin las cuales no habría sido posible llegar al final de esta carrera;

Carlos Rueda Sáenz, José Tomas Valdés, Norberto Sánchez Criado, Alejandro Morán Rueda, y especialmente a José Antonio Ramos (Padre e hijo).

Gracias amigos.

Resumen.

Hoy en día cualquier parte de un negocio está sujeta a nuevas expectativas, competidores, canales, amenazas y oportunidades. Todo negocio tiene el potencial de ser un negocio digital.

A medida que en nuestro mundo aumenta el número de dispositivos inteligentes y conectados, desde teléfonos hasta automóviles y objetos IOT conectados, las empresas que suministran productos o servicios con instrumentos digitales, las que recogen datos de las interacciones con el mercado y utilizan información para optimizar rápidamente su cadena de valor están adquiriendo una ventaja competitiva.

Las empresas que se transforman digitalmente podrán conectarse más estrechamente con los clientes, acelerar el ritmo de la innovación y, como resultado, obtener una mayor participación en los beneficios de sus sectores. Hoy en día, las empresas transformadas digitalmente tienen una ventaja; mañana, sólo las empresas digitales tendrán éxito.

Compañías como Nike y Burberry han pasado de ser proveedores de bienes de consumo tradicionales a líderes digitales, creando valor a través de productos, servicios y experiencia física y digital.

Son las vidas digitales de los clientes las que están cambiando las reglas de compromiso, y la lealtad del cliente se fortalece para las marcas que siguen el ritmo.

Por ello en este trabajo hemos considerado la opción de desarrollar un software que facilite a estas empresas, poder acceder a de múltiples plataformas de software desde una misma aplicación de escritorio.

Índice general.

Tabla de contenido

Capitulo 1. Introducción y objetivos.

1.1 Introducción.

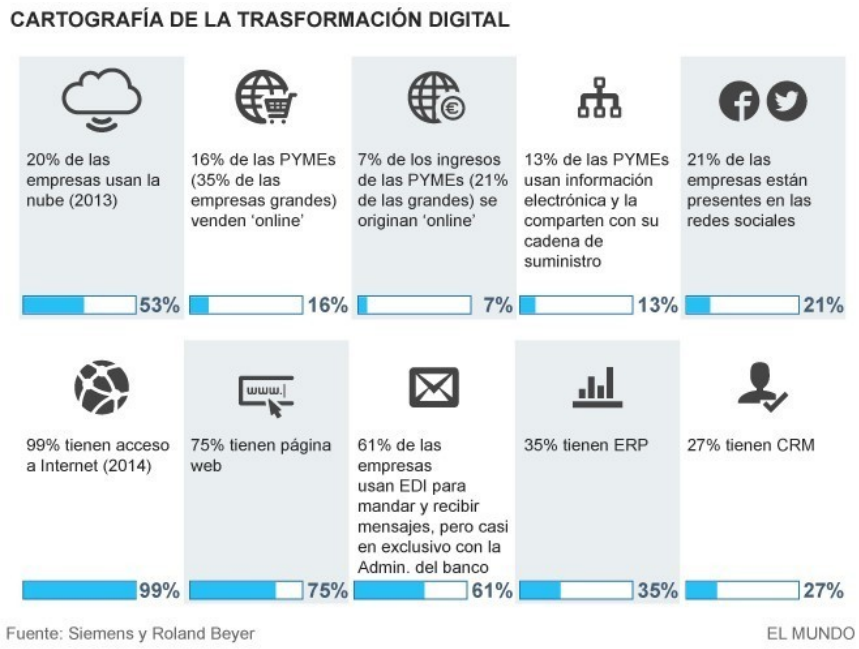
Durante los últimos años hemos podido observar como en todos los sectores se ha llevado a cabo una revolución digital en las empresas. El propio progreso requiere cada vez de procesos mas rápidos y efectivos, la evolución propia de los mercados requiere de sistemas de software para poder registrar los cambios, auditar procesos, calcular diferentes tipos de requerimientos y sobretodo, facilitar el trabajo humano.

Actualmente el 19% de las compañías que cotizan en las principales bolsas del mundo (Ibex, S&P500, Dax…) cuentan con un Chief Digital Officer y según datos que nos muestra thevalley.com esta cifra va en aumento.

También podemos observar cómo tanto la formación cómo los conocimientos en asuntos de digitalización, TIC, y ciberseguridad son cuestiones más valoradas y a tener en cuenta en las juntas directivas y consejos administrativos de todas las empresas.

Por último, es importante definir que la monetización de la transformación digital es un hecho real en todos los sectores ya que ha aumentado los ingresos del 70% de las empresas que se han inmerso en procesos de este tipo.

Tabla 1https://www.elmundo.es/economia/2016/05/23/573e0e7846163f557a8b45eb.html



Por ello nuestro objetivo es poder ofrecer una solución de software empresarial a las empresas. Permitiéndoles tener una suite de plataformas online unificadas en una misma apliacion.

1.2 Objetivos. Unificación de varias nubes en una misma App

El trabajo que se plantea ofrece la posibilidad de que cualquier empresa tenga una aplicación de escritorio gratuita, la cual unificará las principales aplicaciones de ventas y negocio, servicios, analítica de datos, marketing tanto B2B como B2C que una empresa de un tamaño de entre 200 y 5000 empleados pueda necesitar.

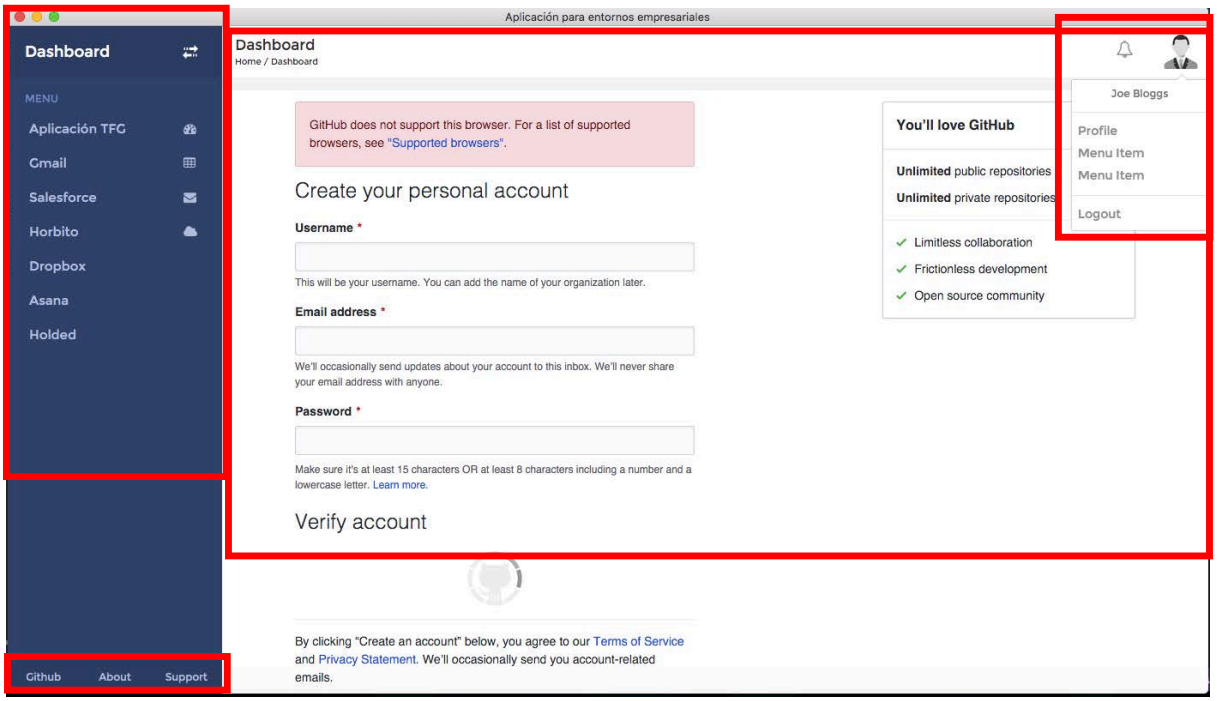


Tabla 2 Captura diseño funcional aplicación

Para ello nos hemos focalizado en las principales herramientas de software empresarial existentes en el mercado, tales como CRM, ERP, SCM, BI …

La empresa que utilice varias aplicaciones en la nube pueda administrar todas estas desde una sola aplicación, además de ofrecérsele la oportunidad de tener un sistema compartido de archivos entre usuarios.

Para poder desarrollar un software con dichas medidas se pueden desarrollar diferentes objetivos intermedios que terminarán convirtiéndose en el enfoque final de nuestra aplicación.

En definitiva, para nuestra aplicación unificadora de diferentes plataformas nos centraremos en las siguientes tareas base.

1. Deberemos enfocarnos en la construcción de una aplicación de escritorio que sea fácil de usar, sobretodo a la hora de cambiar de **aplicación secundaria***

2. La aplicación debe ser segura y sobretodo ágil a la hora de cambiar de diferentes sistemas de software. Es decir, la posibilidad de navegar internamente no debe afectar a la experiencia de usuario.

3. Por último, nos deber permitir trabajar de forma normal, sin cambios de gran contraste sobre las aplicaciones que se estén unificando.

4. Por último y no menos importante es imprescindible definir las aplicaciones que se van a integrar en nuestra aplicación de escritorio.

Hemos considerado de vital importancia (tal y como se podrá observar a lo largo de todo el documento) que la aplicación de escritorio sea modular. Con esto nos referimos a que se pueda elegir las diferentes aplicaciones que se requieran unificar (todas ellas aplicaciones online).

El hecho de que la aplicación sea modular también nos permitirá hacerla escalable a la medida de los clientes que quieran adquirirla, ya que se publicará de forma gratuita en Internet. Por ello se podrá encontrar a lo largo de este documento toda la información sobre la app en detalle con su código comentado. Por último, aclarar que el documento ha sido llevado a cabo siguiendo la metodología de trabajo métrica 3.

1.3 Etapas del proyecto.

Para poder llevar a cabo la realización de este proyecto hemos tenido que centrarnos en diferentes fases. Han sido principalmente seis.

1. Sesiones informativas de documentación en las se trata de informarnos sobre qué productos (aplicaciones en la nube) son claves para construir una buena aplicación de escritorio.

2. Toma de requisitos para poder desarrollar el software requerido

3. Estudio previo de los diferentes softwares empresariales disponibles en el mercado.

4. Programación de la aplicación, estudio de la arquitectura y desarrollo de la misma

5. Mejora de errores que hayan podido surgir durante las fases anteriores.

6. Llevar a cabo la documentación correcta explicando tanto la funcionalidad de la aplicación como su desarrollo, para ello usaremos Métrica 3.

1.4 Estructura del proyecto.

1. **Introducción** y **Objetivos:** Preparación inicial para entender el contexto de este documento. Se definirán los enfoques principales de nuestro proyecto.

2. **Motivación** en la que explicaremos el porqué de este trabajo

3. **Acrónimos y palabras clave** que se explicarán en el proyecto

4. **Estudio de viabilidad del sistema:** Realizaremos un análisis completo sobre las limitaciones del sistema, así como determinar las mejores soluciones posibles de todas las propuestas. Analizaremos también de la misma forma las necesidades que pudiesen tener los clientes.

5. **Gestión del proyecto:** En la que llevaremos a cabo las diferentes fases, análisis de la viabilidad, planificación detallada, ejecución, seguimiento y control y cierre y del proyecto.

6. **Análisis del sistema:** Definiremos los requisitos a tener en cuenta
7. **Diseño del sistema:** Realizaremos un estudio exhaustivo sobre la estructura de la aplicación desarrollada tanto a nivel técnico de arquitectura como a nivel de entorno.

8. **Backup del proyecto**: En el que llevaremos a cabo un plan de validación y se llevará a cabo un testeo de la aplicación desarrollada.

9. **Conclusiones**

10. **Manual de usuario** de la aplicación desarrollada, en la que se hará un enfoque profundo sobre la instalación de la aplicación en diferentes sistemas operativos.

11. **Referencias**

12. **Líneas futuras de investigación**

13. **Anexo**

1.5 Tecnología utilizada.

Para el desarrollo de nuestra aplicación usaremos dos medios, por un lado de software y por otro de hardware al que llamaremos medio físico.

Medio Físico:

- MacBook Pro 15"
- Processor Core i7
- GHz - HDD 500 GB
- RAM 4 GB

Medio Digital:

- MacOS Sierra
- Microsoft office para la documentación
- Snagit Inc Application para las imágenes.

Capitulo 2. Motivación.

El objetivo principal de esta aplicación es el hecho de permitir a los trabajadores de la empresa tener de forma centralizada todas sus herramientas.

Los sistemas actuales hacen difícil que los empleados de una empresa se familiaricen con diferentes aplicaciones. Ejecutarlas, iniciar sesión en cada una de ellas, complican los procesos de trabajo a su vez que quitan tiempo al empleado.

Tener una misma aplicación que ejecute varias herramientas de golpe no solo beneficia al empleado sino también los costes de la empresa.

Tener una única aplicación centralizada nos permitirá desarrollar diferentes “features” que se podrán implementar en un futuro. Saber cuánto tiempo destinan los empleados de una empresa a cada una de sus aplicaciones de negocio será clave para mejorar las necesidades de la misma.

Capitulo 3. Acrónimos y palabras clave.

3.1 Acrónimos y palabas clave.

GUI: (Graphic User interface) Interfaz gráfica de usuario

WYSIWYG: What you see is what you get.

TDD: (Test driven development): Consiste en escribir primero las pruebas, después escribir el código fuente que pase la prueba satisfactoriamente y, por último, re factorizar el código escrito.

API: Application programming interface

SCM: (Supply chain management, SCM)**,** Administración de la cadena de suministro

IDE: Intercambio electrónico de datos.

CRM: Customer relationship management.

POO: Programación orientada a objetos.

DS: Diseño del Sistema.

3.1 Definiciones.

Electron: "Es un framework para JavaScript que permite el desarrollo de aplicaciones enriquecidas de escritorio mediante el uso de tecnologías web. Esta desarrollado por GitHub (lo que garantiza revisiones constantes), es de código abierto y multiplataforma (funciona bajo Linux, Mac y Windows)"

Framework: "Estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software"

Software: Soporte lógico de un sistema informático, que comprende un conjunto de componentes lógicos que permiten realizar al usuario unos trabajos concretos.

ERP: "El término ERP se refiere a Enterprise Resource Planning, que significa "sistema de planificación de recursos empresariales". Estos programas se hacen cargo de distintas operaciones internas de una empresa, desde producción a distribución o incluso recursos humanos."

CRM: "Es una aplicación que permite centralizar en una única Base de Datos todas las interacciones entre una empresa y sus clientes."

BI: Business Intelligence, aplicación que se utiliza para recolectar y analizar los datos de diferentes aplicaciones de software empresarial.

Capitulo 4. Estudio de la viabilidad del sistema.

Establecimiento del alcance del sistema.

En esta parte del documento analizaremos la viabilidad del sistema, viendo todas y cada una de las diferentes partes o etapas de EVS. Es importante recalcar que nuestro sistema no tiene un cliente final que haya pedido una serie de requisitos, sino que se trata de una iniciativa emprendedora en la que se pretenden resolver una serie de problemáticas que las empresas tienen a día de hoy.

4.2.1 Estudio de la solicitud.

La aplicación se ha desarrollado con la misión de ofrecer en primer lugar rapidez al empleado a la hora de iniciar todos sus procesos, o cambiar de aplicación. Por otro lado deberá también

reducir costes a las empresas aparte de ofrecer una analítica de datos sobre las aplicaciones de software empresarial que más estén usando en sus procesos del día a día.

Nuestra plataforma deberá ofrecer al usuario la posibilidad de manejar archivos, procesos complejos de negocio y diferentes características divididas en las diferentes aplicaciones que la compongan.

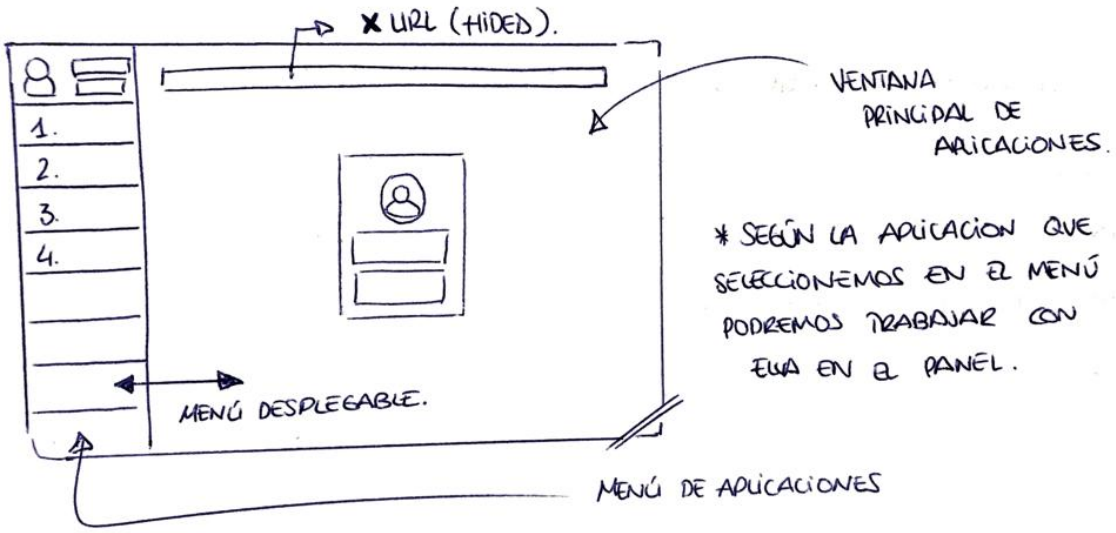
- Posibilidad de enviar y recibir correos de forma segura mediante correo.
- Chatear en tiempo real gracias a aplicaciones de mensajería instantánea.
- Posibilidad de realizar diferentes funcionalidades de negocio por parte de la aplicación Holded.
- Subir y almacenar archivos en la nube de forma segura y mediante un cifrado SHA256 (Horbito)
- Realizar procesos de ventas (Salesforce)
- Realizar procesos de servicios y de facturación.
- Consultar cualquier tipo de dato o archivo al que se tenga permiso.

El cliente será un usuario único al que se le podrá adjudicar diferentes permisos y accesos a las aplicaciones que el administrador de sistemas de la compañía que lo impleante decida darle permisos.

El proyecto está liderado por **Israel Gonzalez Carrasco** y será dirigdo por **Pablo Camprubí Garcia**. En el documento se hará alusión a personas externas o subcontratados para desarrollar el sistema y llevar a cabo las tareas correspondientes.

4.2.2 Identificación del alcance del sistema

Con el desarrollo de la aplicación se deberá llevar una interfaz de usuario sencilla de utilizar, amigable y predictiva. Cualquier usuario con conocimientos informáticos medios tiene que ser capaz de poder moverse a través de ella. A continuación, se adjunta un “Sketch” o boceto sobre como debería visualizarse.



En el menú de aplicaciones tendremos mediante botones acceso a las diferentes aplicaciones. En el panel central desplegaremos la visualización de la interfaz del software elegido.

El sistema debe ser capaz de ejecutarse en cualquier Ordenador independientemente del sistema operativo que tenga instalado. Al tratarte de un conjunto de aplicaciones hemos decidido que todas ellas sean de primera calidad, servicios profesionales utilizados por otras empresas que sean referencias en su industria.

Nuestra aplicación cumple con las Heurísticas de Nielsen y con muchas otras especificadas estudiadas por compañías lideres como Google o Salesforce lideres en el desarrollo de software.

Hemos procurado que el sistema obviamente use el lenguaje de los usuarios, por eso hemos procurado que las aplicaciones (internas) puedan también ser configuradas en diferentes idiomas.

Deberán proporcionar todas ellas una consistencia y estándares mínimos para que no haga falta preguntar sobre el propio sistema.

Tanto nuestro marco de aplicación como las incluidas deben ser intuitivas.

4.2.3 Identificación de los interesados del sistema (StakeHolders)

Puesto que nuestros Stakeholders serán compañías los hemos dividido en diferentes tipos.

Al ser una aplicación instalable únicamente por licencia de sistema operativo (es decir se puede instalar máximo una por cada ordenador) la consideramos completamente escalable. Por eso mismo no influye el tamaño de la empresa en este sentido.

Según el sector al que pertenezca la empresa puede estar interesada en cambiar el tipo de aplicaciones que vienen por defecto. Por ello hemos incluido una serie de aplicaciones posibles secundarias que mostramos a continuación.

Para empresas focalizadas en el **sector de las ventas:**

- Demio
- Hubspot
- Zoho CRM

Para empresas focalizadas en el **sector de servicios:**

- Kolobri App
- Samp
- Wharf

Para empresas focalizadas en el **sector financiero:**

- Cromberg
- Mercury
- Budgie

Para empresas focalizadas en el **sector hotelero:** (Actualmente no podemos recomendar aplicaciones específicas)

Para empresas focalizadas en el **sector Sanitario:** (Actualmente no podemos recomendar aplicaciones específicas)

Para empresas focalizadas en el **sector construccion:** (Actualmente no podemos recomendar aplicaciones específicas)

Para empresas focalizadas en el **sector Telecom:**

- Dixa
- Slack (Communication)
- uPhone

Para empresas focalizadas en el **sector de Inmobiliario:**

- ZuZu
- Zoho CRM
- Holded

4.2 Estudio de la situación actual.

4.3.1 Valoración del estudio de la situación actual

A continuación, realizaremos un estudio del funcionamiento actual de algunos sistemas y aplicaciones informáticas en las empresas actuales. Por lo general y debido a la digitalización de las empresas en los últimos años han surgido multitud de plataformas que intentan ayudar a las empresas a mejorar sus procesos. Dado al "boom" que se está dando actualmente sobre la digitalización de las empresas hace que nuestro servicio deba ser mucho más competente.

Desde este trabajo no se pretende estudiar al resto de plataformas que ofrezcan servicios parecidos como si se tratase de competencia sino cómo una ayuda y oportunidad de la que se pueden sacar ideas para ofrecer un mejor servicio al cliente.

Empresas tecnológicas como Google, Oracle, Salesforce, Sage, Sap etc… ofrecen a nuestra aplicación la oportunidad de no tener que desarrollar el software especifico para diferentes procesos de negocio, sino que, nos permiten de hecho poder centrarnos en mejorar su utilidad e integración.

4.3.1 Realización del diagnostico. → Ultimo párrafo incluir cosas de tryshift.

Tras "realizar la valoración del estudio de la situación actual, se puede observar como a día de hoy existe la necesidad real de implementar una aplicación como la que se presenta en este proyecto".

Si nos centramos en los softwares y plataformas actuales que podrían competir con nosotros, muy pocos ofrecen la posibilidad de elegir las plataformas con las que el usuario realmente desea trabajar por lo que consideramos que estamos desarrollando un programa pionero en el mercado.

Si no centramos en una plataforma que pueda hacer competencia a nuestra aplicación tras un estudio de mercado a través de internet tan solo hemos encontrado una llamada tryshift que es relativamente nueva, por lo que podemos asegurar que es una idea de mercado relativamente nueva.

4.3 Definición de los requisitos del sistema.

Como hemos podido observar en los apartados anteriores hemos llevado a cabo un estudio del estado de la solicitud presentado por el cliente, y a su vez hemos llevado a cabo un estudio de la situación actual de mercado junto con el alcance que podría llegar a tener nuestro sistema. Por ello en esta sección llevaremos a cabo una definición clara y exhaustiva en la que precisará sobre la definición de los requisitos que nuestro sistema deba cumplir para contemplar las expectativas del cliente.

El equipo de desarrollo llevará a cabo las tareas necesarias para cumplir con las expectativas de producto. La lista de requisitos nos proporcionará una visión global sobre las funcionalidades principales de la plataforma aplicación que llevemos a cabo.

4.4.1 Identificación de los requisitos del sistema.

Es de vital importancia dividir nuestros requisitos en diferentes apartados. Facilitará tanto al cliente final como a los propios desarrolladores y programadores saber en que partes y secciones del documento centrarse en caso de modificaciones (totalmente posibles ya que a lo largo del documento se pueden dar).

Distinguiremos las siguientes categorías dentro de los requisitos del sistema

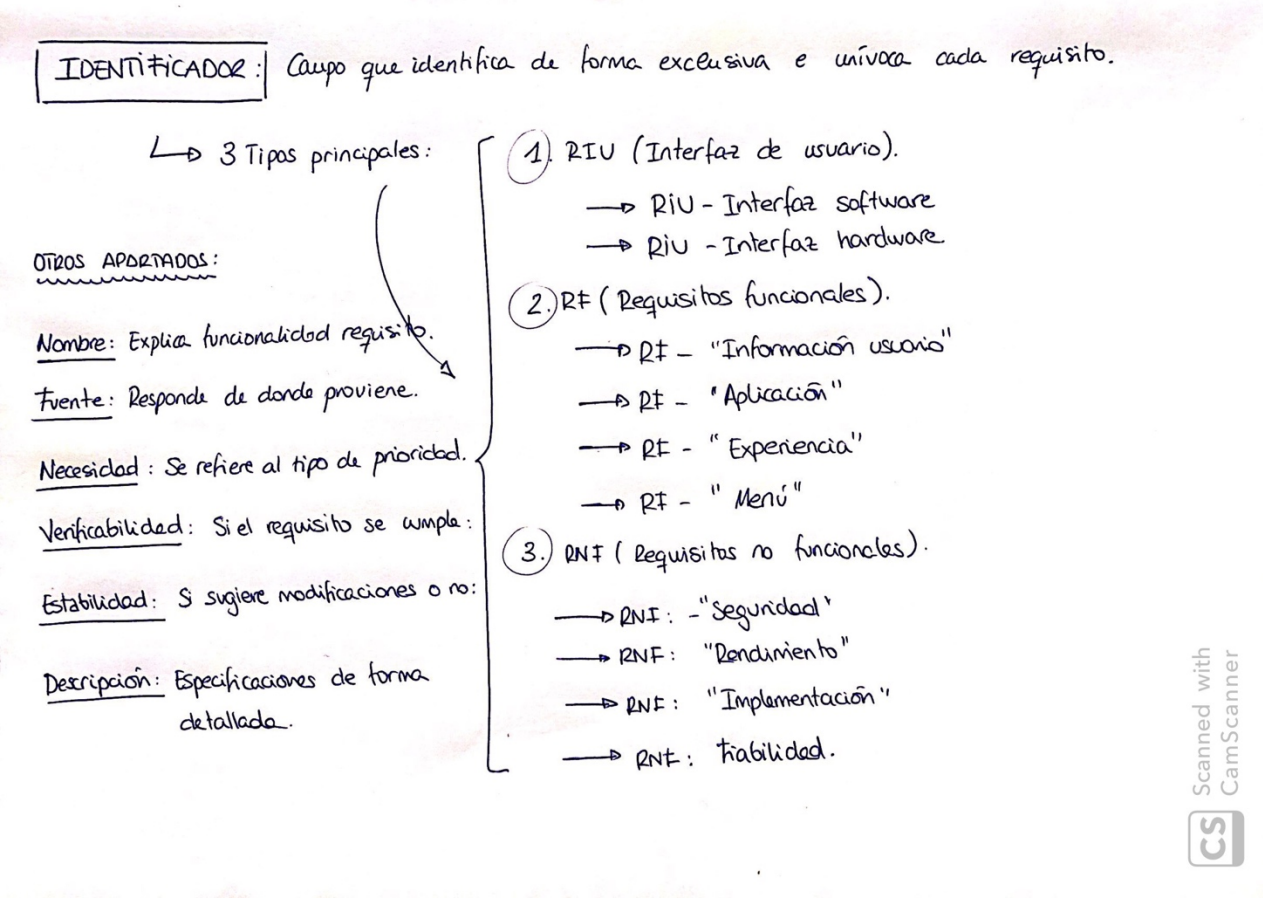
Requisitos de interfaz de usuario:	Forma en la aplicación se comunica con el usuario final.
Requisitos funcionales:	Aquellos que debe cumplir el sistema / aplicación
Requisitos no funcionales:	Aquellos que ofrecen la solución al problema.

En este documento hemos llevado a cabo la descripción de los requisitos de usuario con el formato que se presenta en la imagen inferior, siguiendo siempre los estándares de IEEE.

Identificador: Identificación única del requisito
Nombre: Nombre del requisito
Prioridad: 3 tipos → Muy elevada,alta, media o baja
Fuente: Cliente o equipo de desarrollo
Necesidad: Primordial, opcional, baja necesidad
Verificabilidad: Alta, media o baja → Si se puede verificar que el requisito se cumple.
Estabilidad: si el requisito se modifica con el paso del tiempo
Descripción: En la descripción se explica de la forma adecuada detallada y ordenada el requisito en concreto.

A continuación, incluiremos una página en la que se muestra un workshop que se llevó a cabo con el equipo de desarrollo de producto para identificar los diferentes tipos secciones dentro de la tabla de requisitos.

Veamos cómo hemos llevado a cabo la nomenclatura.



4.4.2 Obtención de los requisitos del sistema.

4.4.2.1 Requisitos de la interfaz de usuario.

4.4.2.1 Requisitos de interfaz de software.

Identificador: RIU-INTERFAZ DE SOFTWARE001

Nombre: Lenguaje natural
Prioridad: 3 tipos → Muy elevada.
Fuente: Equipo de desarrollo
Necesidad: Primordial
Verificabilidad: Alta
Estabilidad: Media, puede modificarse según diferentes opciones.
Descripción: La aplicación debe ser legible en todo momento, el usuario debe ser capaz de interaccionar con ella de forma natural. Es recomendable (Incluir para otras versiones) que la aplicación disponga de varias lenguas.

Identificador: RIU-INTERFAZ DE SOFTWARE002

Nombre: Diseño responsivo
Prioridad: 3 tipos → Muy elevada.
Fuente: Equipo de desarrollo
Necesidad: Primordial
Verificabilidad: Alta
Estabilidad: Alta, no es opcional.
Descripción: La aplicación debe ser responsiva, con esto nos referimos que tanto la aplicación principal como el resto de aplicaciones deben seguir unos patrones webs que sean amoldables a diferentes tipos de pantallas y frames con los que se utiliza la app

Identificador: RIU-INTERFAZ DE SOFTWARE003

Nombre: Heurísticas de Nielsen
Prioridad: 3 tipos → Muy elevada.
Fuente: Equipo de desarrollo
Necesidad: Primordial
Verificabilidad: Alta
Estabilidad: Media, puede modificarse según diferentes opciones.
Descripción: La aplicación debe ser legible en todo momento, el usuario debe ser capaz de interaccionar con ella de forma natural. Es recomendable (Incluir para otras versiones) que la aplicación disponga de varias lenguas.

Identificador: RIU-INTERFAZ DE SOFTWARE003

Nombre: Consistencia y estandares
Prioridad: 3 tipos → Muy elevada.
Fuente: Equipo de desarrollo
Necesidad: Primordial

Verificabilidad: Alta
Estabilidad: Media, puede modificarse según diferentes opciones.
Descripción: El usuario debe seguir las normas y convenciones de la plataforma sobre la que está implementando el sistema, para que no se tenga que preguntar el significado de las palabras, situaciones o acciones del sistema.

Identificador: RIU-INTERFAZ DE SOFTWARE003

Nombre: Visibilidad del estado del sistema y estandares
Prioridad: 3 tipos → Muy elevada.
Fuente: Equipo de desarrollo
Necesidad: Primordial
Verificabilidad: Alta
Estabilidad: Media, puede modificarse según diferentes opciones.
Descripción: sistema debe informar a los usuarios del estado del sistema, dando una retroalimentación apropiada en un tiempo razonable.

4.4.2.1 Requisitos de interfaz de hardware.

Identificador: RIU-INTERFAZ DE HARDWARE001

Nombre: Uso del sistema
Prioridad: 3 tipos → alta.
Fuente: cliente
Necesidad: Primordial
Verificabilidad: Alta
Estabilidad: Media, puede modificarse según diferentes opciones.
Descripción: Para interactuar con la aplicación será necesario un sistema informático con un teclado ratón y una pantalla (básicamente un ordenador/portátil).

Identificador: RIU-INTERFAZ DE HARDWARE002

Nombre: Capacidades del sistema
Prioridad: 3 tipos → alta.

Fuente: cliente
Necesidad: Primordial
Verificabilidad: Alta
Estabilidad: Media, puede modificarse según diferentes opciones.
Descripción: Para interactuar con la aplicación será necesario un mínimo de 4gb de RAM de memoria en el sistema en el que se vaya a usar. Esto es debido a que la base de la aplicación es Electron. Una aplicación que requiere de un mínimo de memoria RAM para su correcto funcionamiento.

4.4.2.2 Requisitos funcionales.

Identificador: RF-INFORMACION USUARIO 003

Nombre: Acceso a soporte
Prioridad: 3 tipos → alta.
Fuente: Equipo de desarrollo
Necesidad: Primordial
Verificabilidad: Alta
Estabilidad: Media, puede modificarse según diferentes opciones.
Descripción: Al tratarse de un sistema infinito, dado a que se puede integrar cualquier software empresarial en la nube el usuario deberá poder acceder a un apartado de soporte para modificar las aplicaciones a las que desea conectarse.

Identificador: RF-INFORMACION USUARIO 004

Nombre: Login incorrecto
Prioridad: 3 tipos → alta.
Fuente: Equipo de desarrollo
Necesidad: Primordial
Verificabilidad: Alta
Estabilidad: Media, puede modificarse según diferentes opciones.
Descripción: En el caso de que el usuario no haga el inicio de sesión de forma incorrecta cada uno de los sistemas integrados deberá tomar sus propias medidas. Es

importante recordar que nuestra aplicación tan solo se encarga de facilitar sistemas en la nube de forma on-premise.

Identificador: RF-APLICACION 001

Nombre: Acceso al sistema
Prioridad: 3 tipos → alta.
Fuente: Equipo de desarrollo
Necesidad: Primordial
Verificabilidad: Alta
Estabilidad: Media, puede modificarse según diferentes opciones.
Descripción: Para interactuar con la aplicación será necesario un icono visible en todos los sistemas en los que se instale. Este icono deberá proporcionar acceso a la aplicación pudiendo de este modo cualquier usuario acceder a su espacio personal.

Identificador: RF-MENU001

Nombre: Acceso a aplicaciones en la nube
Prioridad: 3 tipos → alta.
Fuente: Equipo de desarrollo
Necesidad: Primordial
Verificabilidad: Alta
Estabilidad: Media, puede modificarse según diferentes opciones.
Descripción: Para interactuar con las diferentes aplicaciones el usuario deberá poder acceder mediante un panel de menú izquierdo a sus correspondientes instancias de aplicaciones instaladas.

4.4.2.2 Requisitos no funcionales.

4.4.2.2.1 Requisitos de seguridad.

Identificador: RNF-SEGURIDAD 001

Nombre: Privacidad de datos
Prioridad: 3 tipos → alta.
Fuente: Equipo de desarrollo

Necesidad: Primordial
Verificabilidad: Alta
Estabilidad: Media, puede modificarse según diferentes opciones.
Descripción: Se informará al usuario de que nuestra aplicación no se encarga de almacenar ningún tipo de dato personal, sin embargo todos los datos que se almacenen en las aplicaciones integradas con nuestra aplicación cumplirán con su respectiva política de datos.

Identificador: RNF-SEGURIDAD 002

Nombre: Cifrado
Prioridad: 3 tipos → alta.
Fuente: Equipo de desarrollo
Necesidad: Primordial
Verificabilidad: Alta
Estabilidad: Media, puede modificarse según diferentes opciones.
Descripción: Se informará al usuario de que nuestra aplicación no se encarga de cifrar ningún tipo de dato personal, sin embargo todos los datos que se almacenen en las aplicaciones integradas con nuestra aplicación cumplirán con su respectiva política de cifrado de datos en la nube.

Identificador: RNF-SEGURIDAD 003

Nombre: Firewall
Prioridad: 3 tipos → alta.
Fuente: Equipo de desarrollo
Necesidad: Primordial
Verificabilidad: Alta
Estabilidad: Media, puede modificarse según diferentes opciones.
Descripción: Se informará al usuario de que nuestra aplicación no se encarga de bloquear ningún tipo de ataque externo, pero sin embargo si que se notificará al usuario mediante correo electrónico personal, si alguna de las aplicaciones en la nube integradas a nuestro sistema estuviera siendo atacadas. Todas ellas cumplen con mediadas más que seguras y se consideran líderes en la industria en temas de seguridad y protección de datos

4.4.2.2.2 Requisitos de rendimiento.

Identificador: RNF-RENDIMIENTO 001

Nombre: Rendimiento de inicio de sesión
Prioridad: 3 tipos → alta.
Fuente: Equipo de desarrollo
Necesidad: Primordial
Verificabilidad: Alta
Estabilidad: Media, puede modificarse según diferentes opciones.
Descripción: El usuario no debe esperar más de 3 segundos en que la aplicación le permita interactuar con las plataformas integradas.

Identificador: RNF-RENDIMIENTO 002

Nombre: Rendimiento de apliación externa
Prioridad: 3 tipos → alta.
Fuente: Equipo de desarrollo
Necesidad: Primordial
Verificabilidad: Alta
Estabilidad: Media, puede modificarse según diferentes opciones.
Descripción: En el caso de que alguno de los sistemas integrados no sea accesible en ese momento deberá notificarse al usuario.

c

Nombre: Estabilidad de la plataforma
Prioridad: 3 tipos → alta.
Fuente: Equipo de desarrollo
Necesidad: Primordial
Verificabilidad: Alta
Estabilidad: Media, puede modificarse según diferentes opciones.
Descripción: La plataforma ofrecerá una estabilidad verificable, ya que depende solamente del ordenador en cuestión en el que se utilice. Por otro lado se espera un rendimiento más que estable por parte de los sistemas externos integrados

4.4.2.2.3 Requisitos de implementación.

Identificador: RNF-IMPLEMENTACION 001

Nombre: Persistencia de datos
Prioridad: 3 tipos → alta.
Fuente: Equipo de desarrollo
Necesidad: Primordial
Verificabilidad: Alta
Estabilidad: Media, puede modificarse según diferentes opciones.
Descripción: La aplicación se encargará con tal de evitar cualquier problema de sobrecarga de datos de almacenar en cada uno de las aplicaciones externas la informacion que el usuario pretenda almacenar

Identificador: RNF- IMPLEMENTACION 002

Nombre: Sistema operativo
Prioridad: 3 tipos → alta.
Fuente: Equipo de desarrollo
Necesidad: Primordial
Verificabilidad: Alta
Estabilidad: Media, puede modificarse según diferentes opciones.
Descripción: La aplicación se programará con una arquitectura con la cual sea posible su instalación en cualquier sistema operativo

Identificador: RNF- IMPLEMENTACION 002

Nombre: Diseño de sistema responsivo
Prioridad: 3 tipos → alta.
Fuente: Equipo de desarrollo
Necesidad: Primordial
Verificabilidad: Alta
Estabilidad: Media, puede modificarse según diferentes opciones.

Descripción: La aplicación se programará con una tipa de paradigmas de diseño en el que el tamaño de la pantalla no influirá en su funcionamiento.

4.4.2.2.4 Fiabilidad del sistema.

Identificador: RNF- FIABILIDAD 001

Nombre: Control de errores
Prioridad: 3 tipos → alta.
Fuente: Equipo de desarrollo
Necesidad: Primordial
Verificabilidad: Alta
Estabilidad: Media, puede modificarse según diferentes opciones.
Descripción: La aplicación deberá proporcionar una respuesta fiable a todos los usuarios. Exenta de errores.

Capitulo 5. Gestión de proyecto.

5.1 Introducción.

La realización de estos objetivos requiere una planificación sistemática y una aplicación cuidadosa.
El proyecto en general se refiere a un nuevo emprendimiento con un objetivo específico y varía tanto que es muy difícil de definir con precisión, de todas formas las estimaciones que se detallan a continuación se han acercado bastante al proyecto realizado para llevar a cabo el desarrollo del producto.

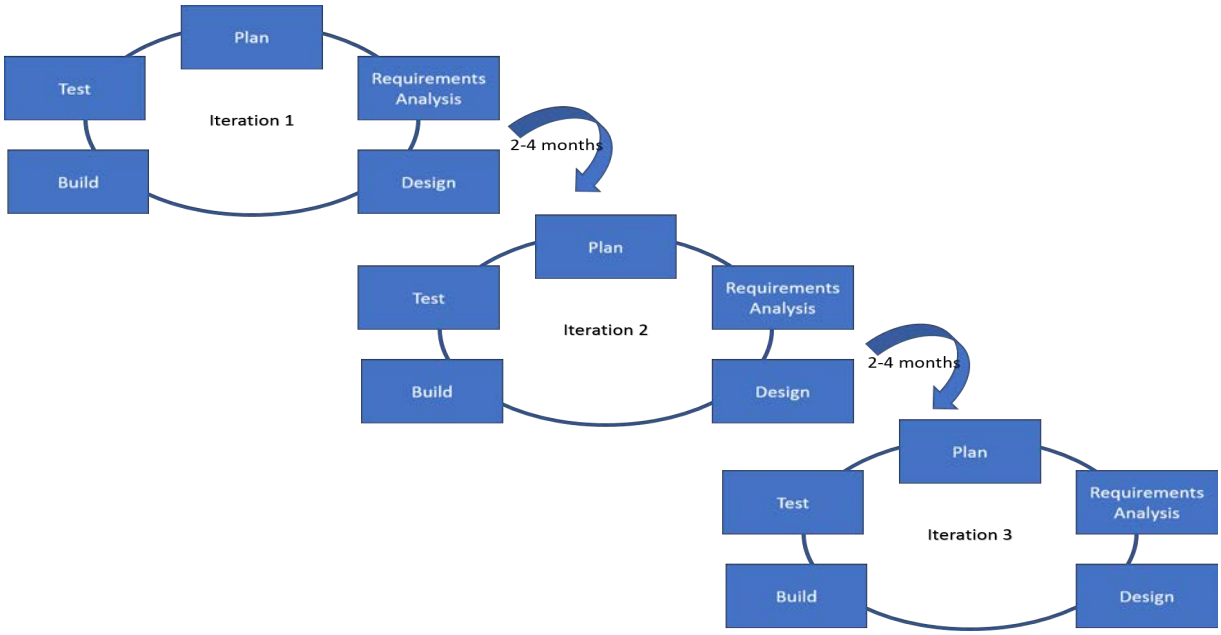
5.2 Ciclo de vida.

Cada proyecto contiene un ciclo de vida en el que se definen las etapas que se van a abordar para llevar a cabo el desarrollo del producto. Dependiendo del tipo de proyecto (también suele influir la industria en la que se desarrolla) su ciclo de vida tendrá un mayor o menor numero de etapas. En cualquier proyecto se tiene como foco principal optimizar los recursos, la forma en la que se utilizan y el tiempo que se invierte en el. Por ello consideramos de vital importancia elegir un ciclo de vida con una serie de fases que se adapten al mismo.

A continuación, veremos los diferentes tipos de ciclos de vida que tiene un proyecto. Todas las definiciones han sido extraídas de la página oficial de preparación para PMP.

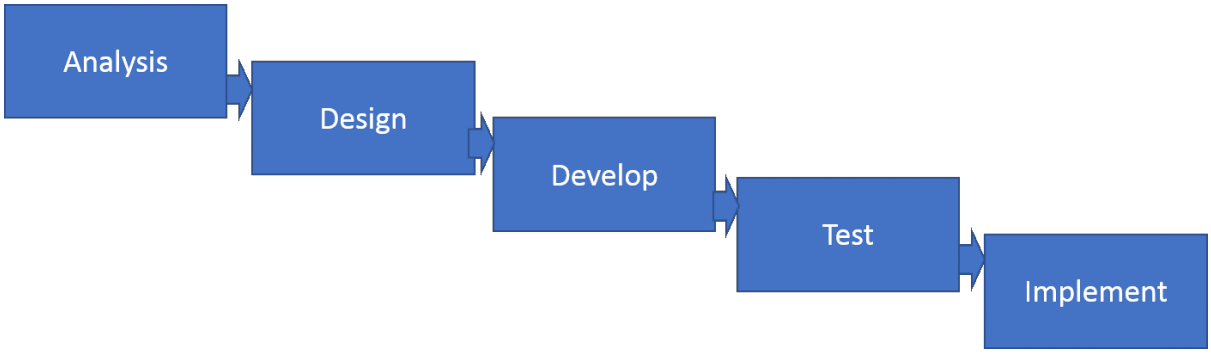
1. **Ciclo de vida iterativo: (seleccionado para el desarrollo)**

A medida que se acortan los plazos de entrega y los requisitos son menos claros, se necesitan enfoques adicionales del ciclo de vida que puedan manejar los cambios de forma más rápida y menos costosa. Encontramos que cuando se dividen los proyectos grandes y complejos en fases más pequeñas (también conocidas como ciclos), estos dan más control (disminución del riesgo y del coste de la revisión).
Como su nombre indica, el proyecto se ejecuta en pequeñas iteraciones, lo que le permite definir mejor los requisitos al inicio de cada ciclo.



2. **Ciclo de vida predictivo:**

Durante los últimos 30 años del último milenio (los'70,'80,'90), el enfoque de Waterfall fue el estándar de oro para todos los proyectos. Se trata de un enfoque totalmente planificado en el que las tres principales limitaciones del proyecto (tiempo, alcance y coste) se determinan a un nivel detallado al comienzo del proyecto. Sólo se necesita saber cuáles son las necesidades y el alcance se fija desde el principio. A continuación, cada fase se presenta de forma secuencial y se gestiona cuidadosamente.



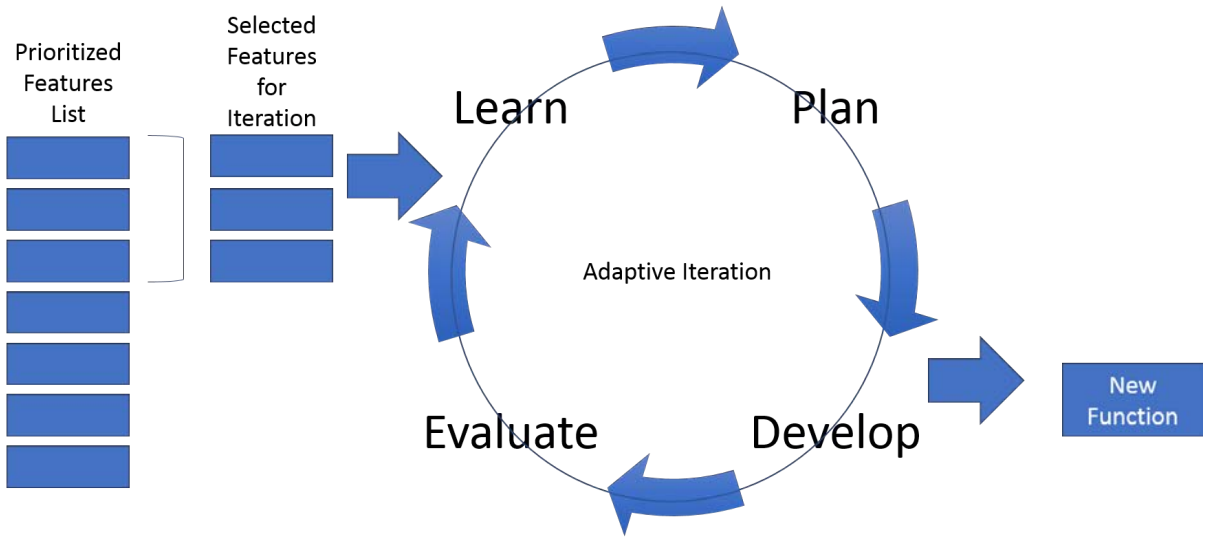
3. Ciclo de vida adaptativo:

"Los ciclos de vida adaptativos, también conocidos como métodos centrados en el cambio o métodos flexibles, responden a altos niveles de cambio y a la participación continua de las partes interesadas.
Existen dos modelos básicos para este tipo de ciclo de vida, los centrados en el flujo (por ejemplo, Kanban) y otros centrados en ciclos iterativos e incrementales (por ejemplo, Scrum)"

Todo el mundo quiere un desarrollo rápido en estos días, así que cuando se necesita ejecutar un proyecto rápidamente, Agile es el camino a seguir. Este enfoque fue creado para manejar los cambios y reducir el riesgo inherente. La Guía del PMBOK® proporciona un apéndice completo (Anexo A3: Visión general de Agile & Lean Frameworks, página 99) sobre el enfoque Agile/Lean. En este tipo de proyectos los equipos entregan las actualizaciones de software en semanas en lugar de meses.

Los proyectos de adaptación son rápidos y están sujetos a dos factores críticos de éxito:

En ellos el cliente debe estar íntimamente involucrado en el proceso y debe poder definir necesidades incrementales al inicio de cada iteración.



FUENTE: http://www.itmplatform.com/en/blog/predictive-or-classical-iterative-or-incremental-and-adaptive-or-flexible-life-cycles/

En este caso hemos decidido escoger el modelo iterativo para desarrollar nuestro proyecto por varias razones que detallamos a continuación.

En primer lugar, por la posibilidad de llevar a cabo los diferentes tipos de análisis de avances, pudiendo así fraccionar el proyecto en diferentes etapas, donde en cada una de ellas el equipo de trabajo tiene la posibilidad de analizar los resultados pudiendo así en cada una de ellas incorporar mejoras para la entrega final.

Por otro lado, se ha hecho especial enfoque también en el manejo de riesgos pues dicho modelo de planificación por secciones también permite la evaluación de riesgos que pueden irrumpir durante la ejecución del proceso.

Por ultimo y debido a la gran importancia que tiene el feedback siento una de las “features” más importantes den los métodos de vida iterativos.

5.3 Organización del proyecto.

A continuación, describiremos cuales son los diferentes tipos de recursos de los cuales vamos a disponer durante la realización del proyecto para el desarrollo del producto. El objetivo es poder planificar y llevar a cabo la mejor estimación posible.

Cómo uno de nuestros objetivos es poder estimar el tiempo que vamos a necesitar para abordar el proyecto de forma completa es necesario anteriormente contabilizar todos los recursos de los cuales disponemos. Con ello nos referimos tanto a los bienes materiales como al capital humano y su experiencia.

5.3.1 Recursos humanos del proyecto

Project Manager	Persona encargada de llevar a cabo la planificación y liderar las diferentes fases del proyecto. Se encarga de tener una relación directa con el cliente así como de dirigir al equipo de desarrollo.
Analista funcional	El analista es la persona encargada de llevar a cabo las definiciones y de ayudar a entender al resto del equipo los requisitos funcionales del cliente. Debe abordar junto con el equipo de desarrollo cómo hacerlo pues se supone que es una persona experimentada.
Diseñador	Se encargará de diseñar la estructura de la aplicación
Arquitecto y desarrollador	En este caso se contará con una persona que tenga ambos perfiles. Comúnmente un desarrollador con experiencia en proyectos de este tipo debería ser capaz de trabajar como arquitecto.

5.3.2 Recursos materiales del proyecto

Software	Al tratarse del desarrollo de una aplicación web necesitaremos las siguientes tecnologías: 1. Navegador web 2. Sublime (editor de texto) 3. Sistema operativo
Hardware	Por otro lado es importante que el ordenador que utilizamos (tal y como se ha llegado a mencionar anteriormente) tenga un mínimo de 4gb de memoria RAM.

Jerarquía de los recursos del proyecto

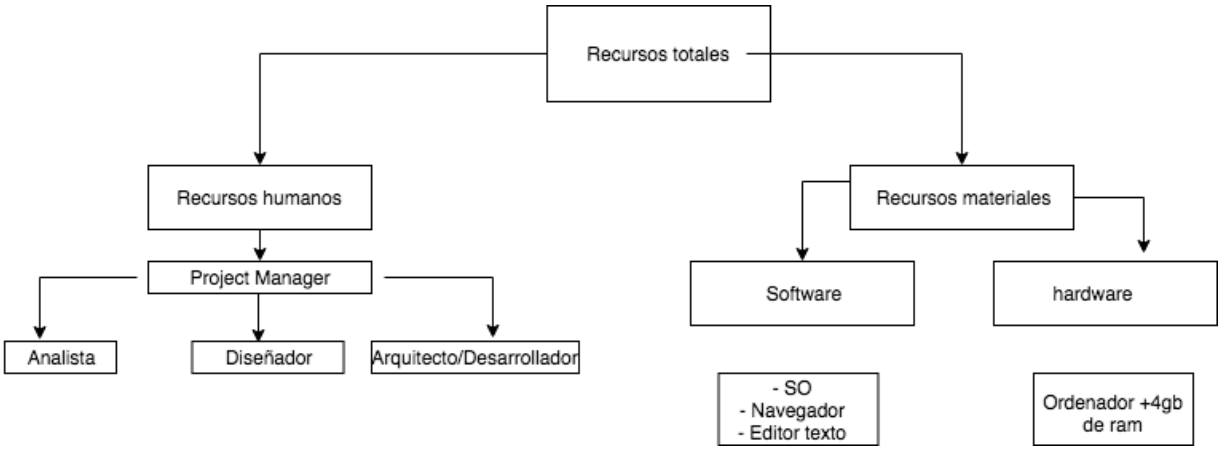


Ilustración 1Recursos totales involucrados en el proyecto

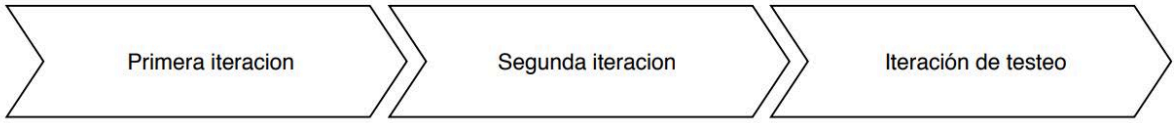
5.3.3. Organización del ciclo de vida.

Tal y como se explicaba anteriormente en este caso tenemos la intención de llevar a cabo un proyecto con ciclos de vida iterativos. Para ello debemos dejar claro el recorrido que este tenga por fases, y las divisiones en iteraciones que tenga.

En este caso se llevarán a cabo 3 iteraciones. La iteración numero uno es en la que se realizará el estudio de la viabilidad del sistema y la correspondiente gestión del proyecto.

La segunda versión será la que sirva como fase de testeo para que los clientes puedan comprobar que los requisitos funcionales declarados se han llevado a cabo en esta fase de implementación.

Por último y la tercera versión de todas servirá para añadir funcionalidades y repasar y testear el desarrollo llevado a cabo.



En cuanto a la primera iteración tal y como se ha comentado anteriormente se llevará a cabo el estudio de viabilidad del sistema. Si nos fijamos en los diferentes apartados que comprende el capitulo 4 podremos distinguir las siguientes fases en la iteración primera.

Estudio de la viabilidad del sistema	**Establecer el alcance completo del sistema**
Estudio de la viabilidad del sistema	Especificar detalladamente los requisitos
Estudio de la viabilidad del sistema	Estudio de las alternativas
Estudio de la viabilidad del sistema	Llevar a cabo un plan cuidadoso teniendo en cuenta los diferentes tipos de costes del proyecto (a ser posible diferenciando entre costes directos e indirectos).
Gestión del proyecto	Ciclo de vida del proyecto
Gestión del proyecto	Organización del proyecto
Gestión del proyecto	Planificación del proyecto
Gestión del proyecto	Costes totales del proyecto

En el caso de que el estudio de la viabilidad del sistema se haga viable se llevará a cabo la gestión del proyecto desde esa misma iteración.

La segunda iteración es en la que se lleva a cabo una mayor parte del desarrollo del proyecto. A continuación, vemos de forma detallada las diferentes partes de la etapa de análisis diseño implementación y pruebas que se desarrollan en esta fase.

Es bastante lógico que esta fase se lleve a cabo tras una revisión por parte del cliente y del equipo técnico de la fase anterior. Por lo general tras la revisión se suelen añadir nuevos cambios o se redefinen algunos requisitos que desde el primer momento podían variar.

Análisis	Alcance
Análisis	Reconocimiento del problema
Análisis	Descripción de la funcionalidad del sistema
Análisis	Especificación de los requisitos
Análisis	Definición de los diferentes casos de uso de la apliación
Análisis	Análisis de las clases para la construcción de la aplicación
Análisis	Descripción de las diferentes interfaces de usuario
Diseño	Alcance y diseño de todo el sistema
Diseño	Definir la arquitectura que tendrá la aplicación
Diseño	Diseño de clases
Diseño	Diseño físico de datos
Diseño	Verificar el modelo de datos y arquitectura.
Implementación del sistema	Llevar a cabo la implementación del sistema
Pruebas	Realizar testeo de pruebas

Por ultimo la iteración de testeo es la que da al producto la fase de vida. En ella será utilizado el producto y se dará feedback al equipo para sus posibles mejoras en caso de que sean necesarias.

5.4 Planificación.

La planificación de proyectos es una disciplina para establecer cómo completar un proyecto dentro de un cierto marco de tiempo, generalmente con etapas definidas y con recursos designados. Una vista de la planificación del proyecto divide la actividad en:

- Fijación de objetivos (que deben ser mensurables)
- Identificación de los productos a entregar
- Planificar el horario
- Haciendo planes de apoyo

Los planes de apoyo pueden incluir los relacionados con: recursos humanos, métodos de comunicación y gestión de riesgos.

La planificación de proyectos de hardware y software en una empresa se realiza a menudo utilizando una guía de planificación de proyectos que describe el proceso que la empresa considera que ha tenido éxito en el pasado.

Las herramientas que se utilizan habitualmente para la parte de programación de un plan incluyen el diagrama de Gantt y el diagrama PERT, que son las que se van a llevar a cabo en este trabajo.

La planificación del proyecto nos informará sobre lo que se necesitamos hacer, quién está trabajando en qué, cómo se va a comunicar el equipo, qué recursos adicionales necesitará y cuándo está previsto que se complete el trabajo. Debido a la variedad de decisiones que se toman durante esta etapa de planificación, es fundamental que no se salte esta parte.

La estructura típica de un plan de proyecto (puede variar ligeramente de un proyecto a otro) incluye las siguientes actividades:

- Establecimiento del título del proyecto
- Aclarar los antecedentes y el contexto del proyecto
- Definición de los objetivos del proyecto
- Organización de tareas
- Selección de los miembros del equipo del proyecto
- Creación de un plan de comunicación
- Identificar los riesgos y planificar para prevenirlos
- Establecimiento del calendario del proyecto
- Decidir el presupuesto del proyecto

- Elección de los recursos del proyecto
- Revisar el plan final y obtener la aprobación de las partes interesadas

Saber qué hacer, cómo se debe completar una tarea y cuándo finaliza el plazo puede ser esencial para la evolución de cualquier proyecto. Planificar antes de pasar a trabajar en una tarea también puede mantener el trabajo de del equipo organizado, esto ayuda a ahorrar tiempo durante la ejecución y reducir los costos del proyecto.

Una planificación apresurada podría afectar evitando que no se reconociesen algunos de los aspectos esenciales. Por ejemplo, la comunicación es uno de los aspectos del proyecto que podría perderse. Con un plan detallado y preciso, cada uno sabe exactamente lo que tiene que hacer y qué procedimientos seguir en caso de que aparezcan errores inesperados.

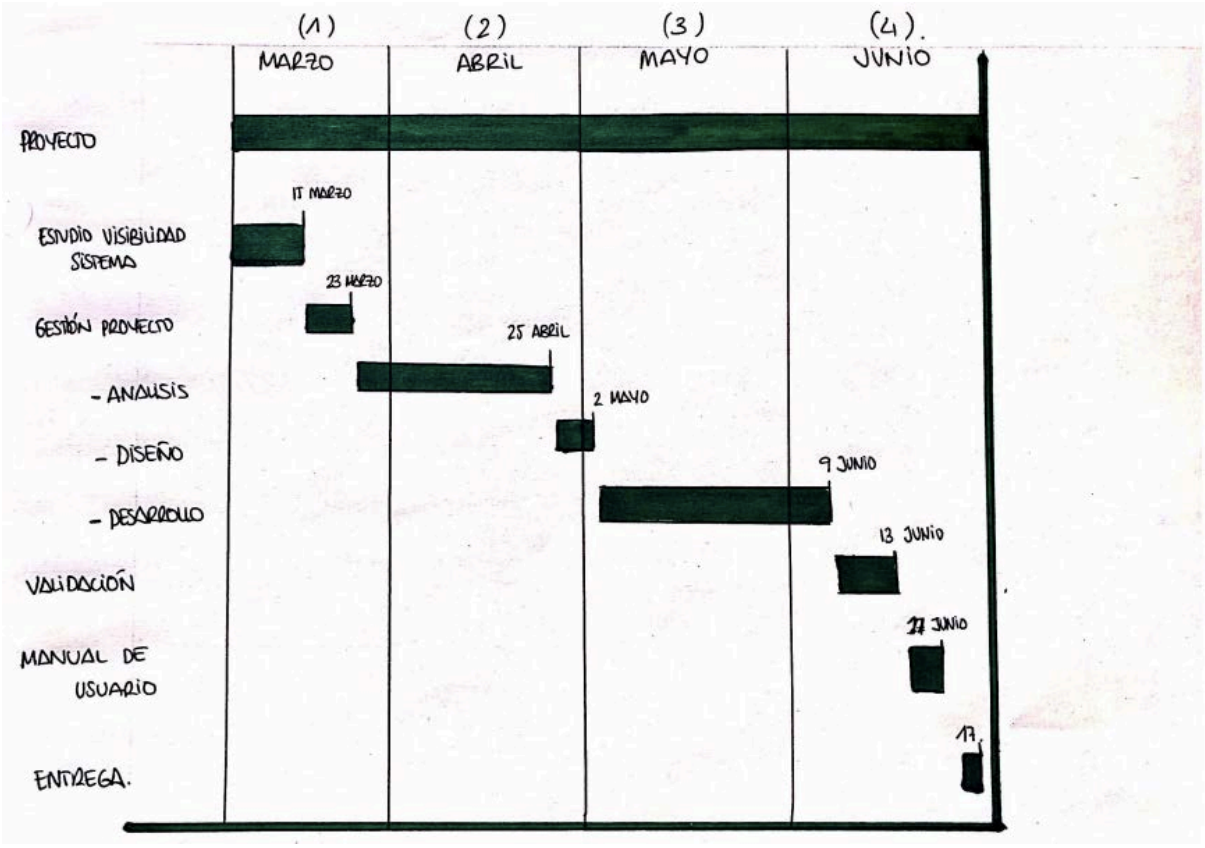
En pocas palabras, el plan de un proyecto debe responder a las siguientes preguntas:

1. ¿Qué tienes que hacer?
2. ¿Cuándo se debe hacer una tarea?
3. ¿Cuándo debe completarse el proyecto?
4. ¿Cuánto costará el proyecto?
5. ¿Quién está a cargo de una tarea específica?
6. ¿Cuáles son las responsabilidades de cada miembro?
7. ¿Qué métodos de comunicación utilizará?
8. ¿Cómo se medirá el estado del proyecto?

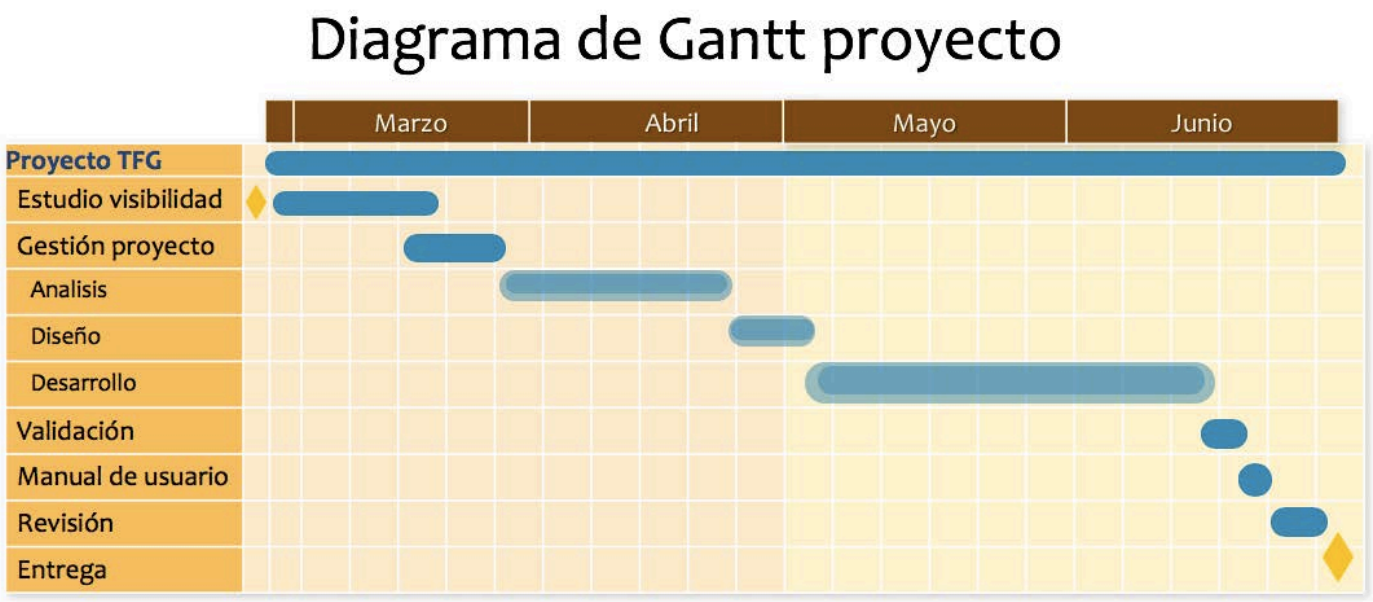
En el proyecto de la aplicación para entorno empresariales se ha llevado a cabo una estimación basada en 100 días. Se ha decidido (al tratarse de un trabajo de fin de grado) empezar precisamente un año antes de la entrega de este proyecto. Es decir, se empezó el 10 de Marzo de 2019 y se plantea finalizarla el 16 de junio de 2019, día de la entrega final.

Tal y como hemos comentado anteriormente se ha decidido utilizar una herramienta para dibujar diagrama de GANTT. Esta se esbozó primero a mano para luego desarrollarla utilizando una plantilla de gantt software.

A continuación, podemos ver la estimación de la planificación del proyecto.

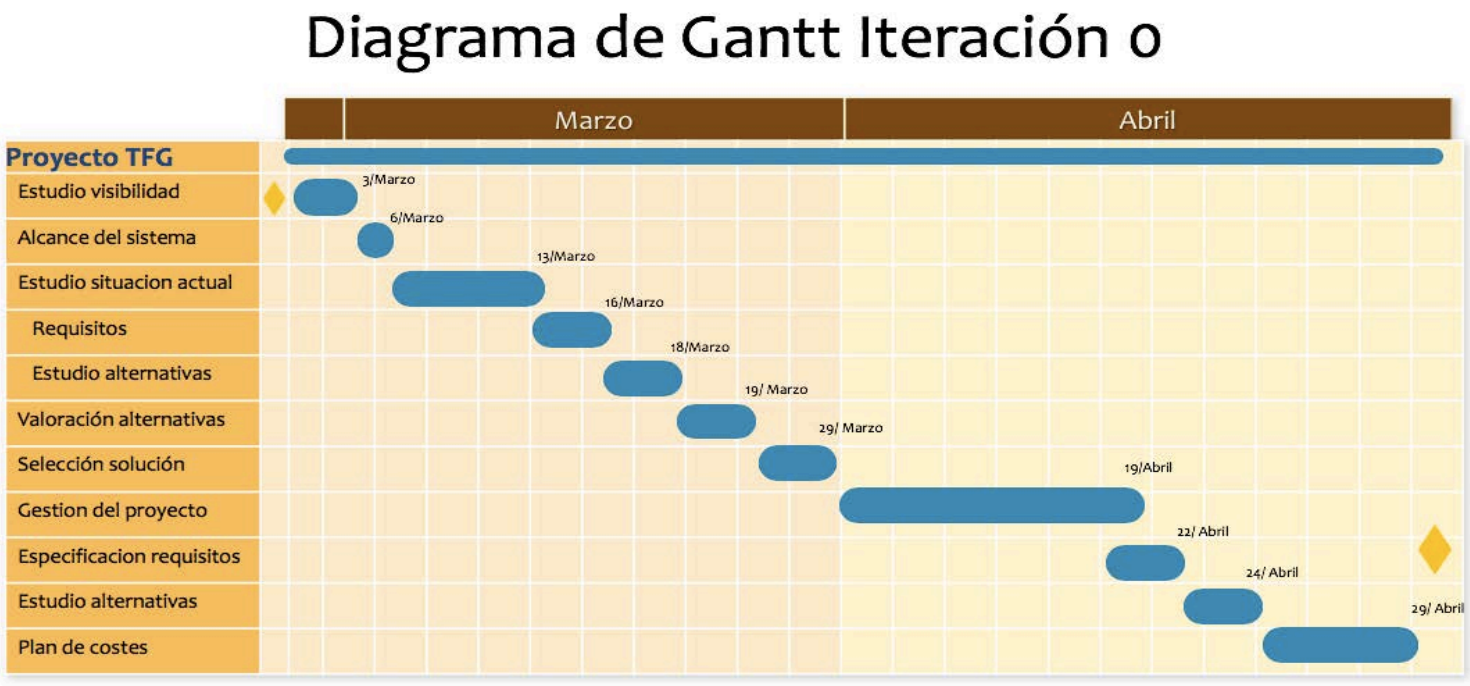


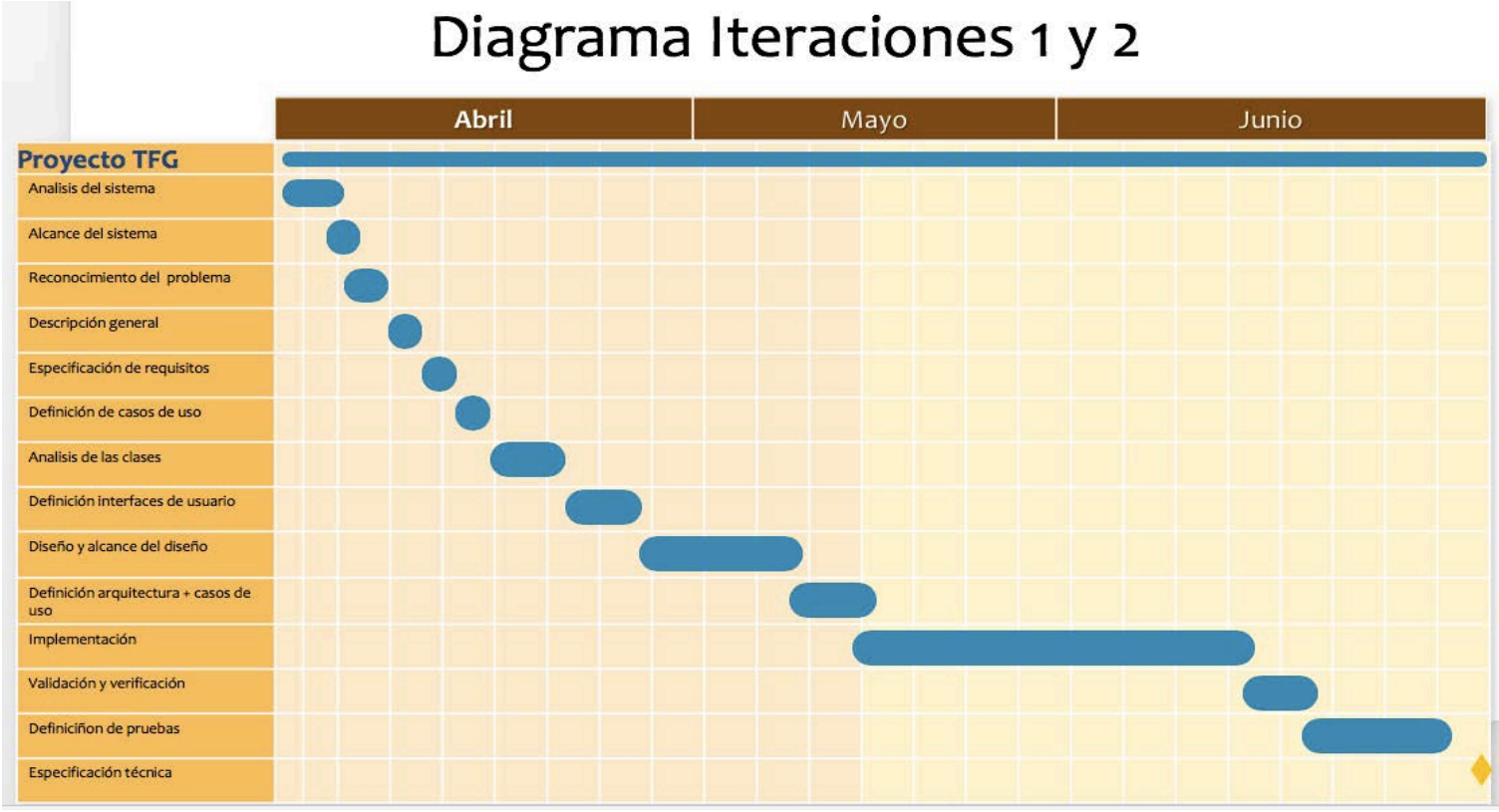
Tras el desarrollo del boceto se decidió pasarlo a un software real para poder enviarlo y documentarlo de mejor manera.



En el apartado del ciclo de vida del proyecto anterior hemos comentado las diferentes iteraciones de las que goza nuestro proyecto. A continuación, también hemos desarrollado un diagrama de Gantt para la primera iteración.

Tal y como se puede ver la fase de Iteración 0 contiene una sobreposición entre la gestión y la especificación de requisitos





5.5 Estimación de costes.

A lo largo del siguiente apartado llevaremos a cabo una estimación de los costes que supone llevar a cabo este proyecto. Como en cualquier tipo de proyecto nos encontramos con diferentes tipos.

Por un lado, tendremos los **costes humanos** o lo que se conoce más bien como costes de personal, los cuales hacen referencia a las personas que se van a contratar para desarrollar el proyecto. Los ingenieros jefes de proyectos diseñadores etc.… todos ellos cuentan con un salario o una nómina los cuales la empresa debe hacerse cargo luego.

A continuación, nos centraremos en cómo se han negociado previamente los salarios otorgados a las diferentes posiciones y roles según las horas.

Role	Nº Horas	Coste/Hora	Coste total
Project Manager	300	30€/hora	9.000€
Analista funcional	240	20€/hora	4.800€
Diseñador	220	15€/hora	3.300€
Arquitecto y desarrollador	300	25€/hora	7.500€
Total	1060 horas totales		24600€

En segundo lugar, tendremos todos los costes relacionados con el hardware informático, es decir, equipos, servidores etc… en este caso no contemplamos más que un solo ordenador que se ha llevado a cabo para terminar el proyecto.

En tercer lugar, se suele tener en cuenta los softwares necesarios para llevar a cabo el desarrollo. En este caso todas las librerías que se han utilizado para programar son libres y el software del ordenador también fue gratuito pues venía previamente instalado con el hardware por lo que el coste total ha sido nulo.

En nuestro proyecto hemos tomado fuertes medidas para mejorar y reducir al máximo los costes. Nos hemos centrado en las siguientes líneas.

- Elegir las herramientas adecuadas para todos los involucrados
- Tener un proceso en marcha, pero mantenerlo simple
- Refinar los flujos de trabajo y el proceso a lo largo del camino
- Medir los esfuerzos y compararlos regularmente

A continuación, mostramos el coste de los recursos materiales los cuales los hemos dividido por hardware como por software.

Hardware	número	Coste	Coste total
Macbook pro 8 pulgadas > 8RAM	1	900€	900€
Raton marca Apple	1	79€	79€
Cascos insonorización para programador	1	160€	160€
			1139€
Software	número	Coste	Coste total
Licencia gratuita de Trello	0	0€	0€
Sistema Operativo Mac Sierra	0	0€	0€
Licencia Office ofimatica	0	0€	0€
Apis NodeJs			0€
Coste total recursos materiales:			1139€

TABLA CON EL CALCULO TOTAL DEL PROYECTO

DESCRIPCIÓN	**TOTAL**
COSTES DE RRHH	24600€
COSTES MATERIALES	1139€
TOTAL, ANTES DE RIESGO	25739€
RIESGO	23165,1€
IVA 21%	29766,6€

El coste total del proyecto asciende a una cantidad total de **29.766,6€**

Capitulo 6. Análisis del sistema.

6.1 Introducción.

El objetivo de este capitulo es poder describir las especificaciones que tiene que cumplir nuestra aplicación. Por entenderlo mejor en este capitulo se definirán y especificarán las necesidades que se desea que cumpla la aplicación. Nuestra aplicación unificadora de escritorio deberá cumplir con las necesidades básicas de las que requiere el usuario.

Antes de describir cómo crear documentación arquitectónica correctamente, necesitamos entender por qué es necesaria.

Hay tres objetivos principales para la documentación arquitectónica:

- Intercambio de conocimientos. Es adecuado para la transferencia de conocimientos entre personas que trabajan en diferentes áreas funcionales del proyecto, así como para la transferencia de conocimientos a nuevos participantes.
- Comunicación. La documentación es el punto de partida para la interacción entre las diferentes partes interesadas. En particular, ayuda a compartir las ideas del arquitecto con los desarrolladores.
- Análisis. La documentación es también un punto de partida para futuras revisiones arquitectónicas del proyecto.

6.2 Alcance del sistema.

Nuestra aplicación de escritorio está desarrollada de tal forma que evita que el usuario deba acceder a un navegador y realizar una navegación entre pestañas para acceder a sus sitios de interés. Nosotros ofrecemos con nuestro proyecto un acceso rápido y seguro a cualquier software empresarial en la nube con el cual podamos trabajar.

Es importante también desde nuestro punto de vista ofrecer una aplicación modular la cual permita al usuario o a la empresa en cuestión cuales son las aplicaciones web a las que se quiere acceder.

Se ha realizado mucho trabajo de investigación para poder descubrir la mejor manera de desarrollar la aplicación. Las sesiones de Brainstorming fueron claves en el desarrollo del proyecto.

6.3 Estándares y restricciones.

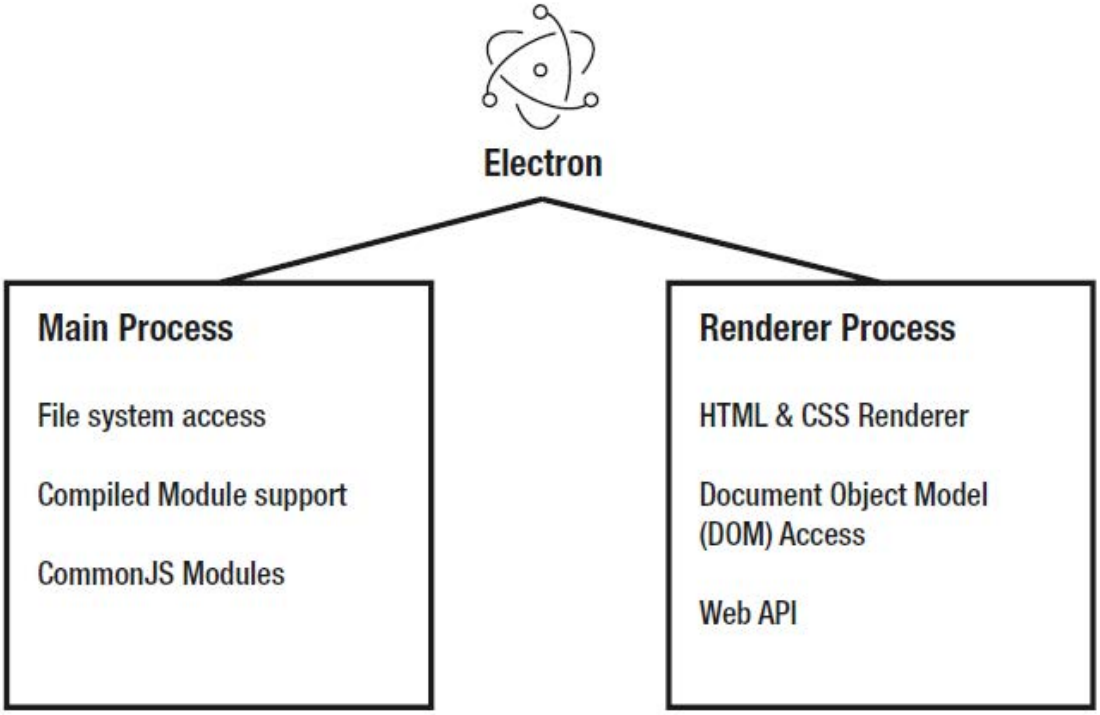
- Toda la documentación que se realice por parte del proyecto deberá seguir métrica 3 como metodología.
- Es necesario contar con acceso a una oficina con internet para desarrollar el trabajo requerido.
- Cada iteración que se defina deberá permitir avanzar sobre la anterior en términos de mejoras de la aplicación
- La aplicación se implementará en español como idioma principal, pero se le perimitrá al usuario acceso al código para cambiarla.
- La aplicación será de código abierto y se publicará a la comunidad para que puedan aportar ideas, programaciones etc…
- Para poder usar la aplicación en correctas condiciones deberemos poder acceder a ella con un ordenador que tenga por lo menos 4 gigas de memoria RAM

6.4 Usuarios principales de la aplicación.

El tipo de usuario que se encargará de utilizar va desde un director de la compañía hasta el becario recién contratado. Nuestra aplicación contiene herramientas de todo tipo y que se usan día a día por cualquier empleado normal. Desde el acceso al correo electrónico hasta el acceso de los reportes de ventas más privados que pueda tener una compañía.

6.5 Entorno operacional.

Nuestra aplicación esta desarrollada a través de electron un Framework basado en nodejs que utiliza una arquitectura de renderización de procesos. Por un lado, tiene un proceso principal y por otro uno renderizado. En la siguiente imagen se puede entender de forma más fácil.



Tal y como podemos ver tenemos un “main process" el cual se encarga de manejar las diferentes actividades que tiene la aplicación tales como eventos del ciclo de vida. Es en este proceso donde encontramos el menú principal de los diferentes procesos que se ejecutan. Por otro lado, tenemos el “proceso renderizado” cuya función principal es encargarse de la user interface del sistema de la aplicación. Cada uno de los procesos renderizados cargará su propio contenido imagen etc. ejecutándolo en un hilo independiente. Tal y como lo podremos comprobar este estará aislado de cualquier interacción con los eventos a nivel de sistema.

En pocas palabras, Electron proporciona un tiempo de ejecución para crear aplicaciones de escritorio con JavaScript puro. La forma en que funciona es - Electron toma un archivo principal definido en su archivo package.json y lo ejecuta. Este archivo principal (normalmente llamado main.js) crea ventanas de aplicaciones que contienen páginas web renderizadas con el poder añadido de interactuar con la interfaz gráfica de usuario (GUI) nativa de su sistema operativo.

En detalle, una vez que se inicia una aplicación utilizando Electron, se crea un proceso principal. Este proceso principal es responsable de interactuar con la GUI nativa de su sistema operativo y crea la GUI de su aplicación (sus ventanas de aplicación).

El mero inicio del proceso principal no ofrece a los usuarios de su aplicación ninguna ventana de aplicación. Estos son creados por el proceso principal en el archivo principal usando algo llamado módulo BrowserWindow. Cada ventana del navegador ejecuta su propio proceso de renderizado. Este proceso de renderizado toma una página web (un archivo HTML que hace referencia a los archivos CSS habituales, archivos JavaScript, imágenes, etc.) y la renderiza en la ventana. Sus páginas web están renderizadas con Cromo por lo que se garantiza un alto nivel de compatibilidad con los estándares.

Por ejemplo, si sólo tuviera una aplicación de calculadora, su proceso principal instanciaría una ventana con una página web donde está su página web real (calculadora).

Aunque se dice que sólo el proceso principal interactúa con la GUI nativa de su sistema operativo, hay técnicas para descargar parte de ese trabajo a los procesos de renderizado (estudiaremos la posibilidad de crear una característica que aproveche dicha técnica).

El proceso principal puede acceder a la GUI nativa a través de una serie de módulos disponibles directamente en Electron. Su aplicación de escritorio puede acceder a todos los módulos del Nodo como el excelente notificador de nodos para mostrar notificaciones del sistema, solicitud de llamadas HTTP, etc.

6.6 Establecimiento de los requisitos de software.

Identificación de los requisitos.

Los requisitos claramente definidos son señales esenciales en el camino que conducen a un proyecto exitoso. Establecen un acuerdo formal entre un cliente y un proveedor de que ambos están trabajando para alcanzar la misma meta. Los requisitos detallados y de alta calidad también ayudan a mitigar los riesgos financieros y a mantener el proyecto dentro de un cronograma. Según la definición del “Business Analysis Body of Knowledge” los requisitos son una representación utilizable de una necesidad.

La creación de requisitos es una tarea compleja, ya que incluye un conjunto de procesos como la obtención, el análisis, la especificación, la validación y la gestión. En este

documento, discutiremos los principales tipos de requisitos para nuestro software y proporcionaremos una serie de recomendaciones para su uso.

Los requisitos del sistema significan una descripción más detallada de los servicios del sistema y de las restricciones operativas, como la forma en que se utilizará el sistema, y de las restricciones de desarrollo, como los lenguajes de programación.

Este nivel de detalle es necesario para aquellos que están involucrados en el desarrollo del sistema, como ingenieros, arquitectos de sistemas, probadores, etc.

Antes de discutir cómo se crean los requisitos, vamos a diferenciar sus tipos.

- Requisitos del negocio. Esto incluye declaraciones de alto nivel de metas, objetivos y necesidades.

- Requisitos de las partes interesadas. También se especifican las necesidades de los grupos de partes interesadas para definir lo que esperan de una solución en particular.

- Requisitos de la solución. Los requisitos de la solución describen las características que debe tener un producto para satisfacer las necesidades de los grupos de interés y de la propia empresa.

1. **Los requisitos no funcionales** describen las características generales de un sistema. También se les conoce como atributos de calidad (aunque este termino se utiliza con menor frecuencia).

2. **Los requisitos funcionales** describen cómo debe comportarse un producto, cuáles son sus características y funciones.

3. **Requisitos de transición.** Un grupo adicional de requisitos define lo que se necesita de una organización para pasar con éxito de su estado actual al estado deseado con el nuevo producto.

Exploremos los requisitos funcionales y no funcionales con mayor detalle. Los requisitos funcionales son características del producto o funciones que los desarrolladores deben implementar para que los usuarios puedan realizar sus tareas. Por lo tanto, es importante dejarlas claras tanto para el equipo de desarrollo como para las partes interesadas. Generalmente, los requisitos funcionales describen el comportamiento del sistema bajo condiciones específicas. Por ejemplo:

“Una función de búsqueda permite al usuario buscar entre varias facturas si desea abonar una factura emitida.”

Aquí hay otro ejemplo simple: “Como huésped, quiero un sofá en el que pueda dormir toda la noche”

Los requisitos suelen estar escritos en texto, especialmente para proyectos ágiles. Sin embargo, también pueden ser visuales

Hemos llevado a cabo la siguiente clasificación de los requisitos:

1. **Requisitos funcionales:** Son los encargados de definir las diferentes funcionalidades de la aplicación.
2. **Requisitos de rendimiento:** Son los encargados de definir los valores de las diferentes variables de rendimiento de la aplicación
3. **Requisitos de operación:** Son los encargados de definir los diferentes niveles de servicio mínimo que debe cumplir la aplicación.
4. **Requisitos de interfaz:** Son los encargados de definir la interacción entre usuario y plataforma.
5. **Requisitos de seguridad:** Estos vienen definidos por las tres reglas básicas del mundo de la seguridad informática. Disponibilidad, confidencialidad e integridad.
6. **Requisitos de calidad:** La aplicación debe cumplir con los estándares de calidad

La definición de cada uno de los requisitos se recoge en la siguiente tabla como ejemplo siguiendo como siempre el estándar IE3:

Identificador: Identificación única del requisito
Nombre: Nombre del requisito
Prioridad: 3 tipos → Muy elevada, alta, media o baja
Fuente: Cliente o equipo de desarrollo
Necesidad: Primordial, opcional, baja necesidad
Verificabilidad: Alta, media o baja → Si se puede verificar que el requisito se cumple.
Estabilidad: si el requisito se modifica con el paso del tiempo
Descripción: En la descripción se explica de la forma adecuada detallada y ordenada el requisito en concreto.

Tabla 3: Tabla de requisitos de software

- Requisitos funcionales:

Identificador: RF01.
Nombre: Acceso a la aplicación.
Prioridad: Muy elevada
Fuente: Equipo de desarrollo
Necesidad: Primordial,
Verificabilidad: Alta.
Estabilidad: Estable
Descripción: El acceso a nuestra aplicación se permitirá mediante un icono en el escritorio. Este icono se debe proporcionar en el escritorio de cualquier sistema operativo ya sea Linux, Windows o Mac.

Identificador: RF02.
Nombre: Navegación.
Prioridad: Muy elevada
Fuente: Equipo de desarrollo
Necesidad: Primordial,
Verificabilidad: Alta.
Estabilidad: Estable
Descripción: El usuario deber poder navegar entre pestañas. Cada aplicación ofrecerá un acceso único. El usuario deber estar registrado en el sistema que vaya autilizar para poder acceder a cualquiera de ellas.

Identificador: RF03.
Nombre: Soporte.
Prioridad: Media
Fuente: Equipo de desarrollo
Necesidad: Opcional
Verificabilidad: Alta.
Estabilidad: Estable
Descripción: El usuario deber poder encontrar un sistema de soporte donde preguntar cualquier tipo de duda o abrir casos. Tal y como hemos comentado anteriormente el producto será de código libre así que podrá preguntar en la comunidad de Github donde se publique el proyecto.

Identificador: RF04.
Nombre: Registro en el sistema externo.
Prioridad: Media
Fuente: Equipo de desarrollo
Necesidad: Opcional
Verificabilidad: Alta.

Estabilidad: Estable
Descripción: El usuario deber poder registrarse en el sistema externo en caso de que no tenga acceso a la plataforma correspondiente. Debe poder acceder a cualquier sección de la aplicación web a través de la aplicación.

Identificador: RF05.
Nombre: Soporte de actualizaciones.
Prioridad: Alta
Fuente: Equipo de desarrollo
Necesidad: Primordial
Verificabilidad: Alta.
Estabilidad: Estable
Descripción: La aplicación debe poder soportar cualquier tipo de actualización que lleven a cabo sus aplicaciones secundarias.

Identificador: RF06.
Nombre: Idioma de la plataforma.
Prioridad: Alta
Fuente: Equipo de desarrollo
Necesidad: Primordial
Verificabilidad: Alta.
Estabilidad: Estable
Descripción: La aplicación debe estar en castellano y se le debe dar al usuario la posibilidad de

Identificador: RF07.
Nombre: Mensaje de error.
Prioridad: Alta
Fuente: Equipo de desarrollo
Necesidad: Primordial
Verificabilidad: Alta.
Estabilidad: Estable

Descripción: En cuanto el usuario cometa errores se le debe transmitir por mensaje.

Identificador: RF08.
Nombre: Información y manual.
Prioridad: Alta
Fuente: Equipo de desarrollo
Necesidad: Primordial
Verificabilidad: Alta.
Estabilidad: Estable
Descripción: La propia aplicación tiene una pestaña desde la cual el usuario podrá acceder a revisar información sobre el sistema desarrollado.

- Requisitos de rendimiento:

Identificador: RR01.
Nombre: Disponibilidad de aplicaciones secundarias o sistemas externos.
Prioridad: Alta
Fuente: Equipo de desarrollo
Necesidad: Primordial
Verificabilidad: Alta.
Estabilidad: Estable
Descripción: Es completamente necesario tener una disponibilidad de aplicaciones secundarias o sistemas externos, en el caso de que los servidores de las aplicaciones a las que nos conectamos no funcionen nuestra aplicación carecería de sentido.

Identificador: RR02.
Nombre: Requisitos de hardware.
Prioridad: Media
Fuente: Cliente
Necesidad: Primordial

Verificabilidad: Alta.
Estabilidad: Estable
Descripción: Es recomendable como ya se ha mencionado en otras ocasiones que los equipos en los que se instale la aplicación tengan como minimo4gb de memoria RAM.

Identificador: RR03.
Nombre: Requisitos de hardware.
Prioridad: Media
Fuente: Cliente
Necesidad: Primordial
Verificabilidad: Alta.
Estabilidad: Estable
Descripción: Es necesario para el funcionamiento de la aplicación que el equipo tenga una conexión correcta a internet.

- Requisitos de interfaz:

Identificador: RI01.
Nombre: Requisitos de software.
Prioridad: Media
Fuente: Cliente
Necesidad: Primordial
Verificabilidad: Alta.
Estabilidad: Estable
Descripción: La aplicación debe poder utilizarse en cualquier ordenador independientemente del sistema operativo que tenga instalado. Las pruebas se realizarán en un Mac pero se espera que funcione en Linux y Windows a su vez.

Identificador: RI02.
Nombre: Menu principal.

Prioridad: Media
Fuente: Cliente
Necesidad: Primordial
Verificabilidad: Alta.
Estabilidad: Estable
Descripción: La aplicación debe contar un menú principal que además sea desplegable. Este menú permite navegar entre las diferentes aplicaciones exteneras.

Identificador: RI03.
Nombre: Acceso a secciones.
Prioridad: Media
Fuente: Cliente
Necesidad: Primordial
Verificabilidad: Alta.
Estabilidad: Estable
Descripción: Cada aplicación debe contar con una actividad y pestaña única para un correcto uso de ellas .

Identificador: RI05.
Nombre: Heurísticas de Nielsen.
Prioridad: Media
Fuente: Cliente
Necesidad: Primordial
Verificabilidad: Alta.
Estabilidad: Estable
Descripción: La interfaz de la aplicación debe contar en todo momento con las heurísticas de Nielsen.

Identificador: RI06.
Nombre: Responsive.
Prioridad: Media

Fuente: Cliente
Necesidad: Primordial
Verificabilidad: Alta.
Estabilidad: Estable
Descripción: La interfaz de la aplicación debe ser completamente responsiva a si como las aplicaciones externas que se le añadan. Esto permitirá una correcta visualización para el usuario

Identificador: RI07.
Nombre: Diseño claro.
Prioridad: Media
Fuente: Cliente
Necesidad: Primordial
Verificabilidad: Alta.
Estabilidad: Estable
Descripción: La interfaz de la aplicación debe contener un diseño claro y legible en todo momento, cumpliendo los patrones de diseño web básicos.

Identificador: RI08.
Nombre: Lenguaje de usuario.
Prioridad: Alta
Fuente: Cliente
Necesidad: Primordial
Verificabilidad: Alta.
Estabilidad: Estable
Descripción: La aplicación deberá contener en todo momento un lenguaje claro con mensajes completamente legibles. Deben hacer la plataforma sencilla e intuitiva de utilizar.

Identificador: RI09.
Nombre: Diseño minimalista.
Prioridad: Media

Fuente: Cliente
Necesidad: Primordial
Verificabilidad: Alta.
Estabilidad: Estable
Descripción: La aplicación deberá contener en todo momento un diseño claro para hacer la navegación fácil para ofrecer la mejor usabilidad posible.

- Requisitos de operación:

Identificador: RO01
Nombre: Tiempo
Prioridad: 3 tipos → Muy elevada, alta, media o baja
Fuente: Cliente o equipo de desarrollo
Necesidad: Primordial, opcional, baja necesidad
Verificabilidad: Alta, media o baja → Si se puede verificar que el requisito se cumple.
Estabilidad: si el requisito se modifica con el paso del tiempo
Descripción: La unidad de tiempo que utilizaremos para medir los tiempos de respuesta de la aplicación serán los segundos.

Identificador: RO02
Nombre: Lenguaje
Prioridad: 3 tipos → Muy elevada, alta, media o baja
Fuente: Cliente o equipo de desarrollo
Necesidad: Primordial, opcional, baja necesidad
Verificabilidad: Alta, media o baja → Si se puede verificar que el requisito se cumple.
Estabilidad: si el requisito se modifica con el paso del tiempo
Descripción: El idioma de la aplicación será el castellano

Identificador: RO03
Nombre: Acceso
Prioridad: 3 tipos → Muy elevada, alta, media o baja

Fuente: Cliente o equipo de desarrollo
Necesidad: Primordial, opcional, baja necesidad
Verificabilidad: Alta, media o baja → Si se puede verificar que el requisito se cumple.
Estabilidad: si el requisito se modifica con el paso del tiempo
Descripción: El acceso a la aplicación es completamente libre para cualquier usuario siempre y cuando haya iniciado sesión en el ordenador en el que se encuentre instalada la aplicación.

Identificador: RO04
Nombre: Base de datos
Prioridad: 3 tipos → Muy elevada, alta, media o baja
Fuente: Cliente o equipo de desarrollo
Necesidad: Primordial, opcional, baja necesidad
Verificabilidad: Alta, media o baja → Si se puede verificar que el requisito se cumple.
Estabilidad: si el requisito se modifica con el paso del tiempo
Descripción: Las bases de datos de los sistemas externos deben estar siempre disponibles.

Identificador: RO05
Nombre: Base de datos
Prioridad: 3 tipos → Muy elevada, alta, media o baja
Fuente: Cliente o equipo de desarrollo
Necesidad: Primordial, opcional, baja necesidad
Verificabilidad: Alta, media o baja → Si se puede verificar que el requisito se cumple.
Estabilidad: si el requisito se modifica con el paso del tiempo
Descripción: Las bases de datos de los sistemas externos deben estar siempre disponibles.

Identificador: RO06
Nombre: Compatibilidad

Prioridad: 3 tipos → Muy elevada, alta, media o baja
Fuente: Cliente o equipo de desarrollo
Necesidad: Primordial, opcional, baja necesidad
Verificabilidad: Alta, media o baja → Si se puede verificar que el requisito se cumple.
Estabilidad: si el requisito se modifica con el paso del tiempo
Descripción: El sistema será únicamente compatible a través de un acceso por parte de un sistema operativo instalado en un ordenador. No soporta móviles ni tablets.

- Requisitos de recursos:

Identificador: RR01
Nombre: Equipo técnico para modificaciones
Prioridad: 3 tipos → Muy elevada, alta, media o baja
Fuente: Cliente o equipo de desarrollo
Necesidad: Primordial, opcional, baja necesidad
Verificabilidad: Alta, media o baja → Si se puede verificar que el requisito se cumple.
Estabilidad: si el requisito se modifica con el paso del tiempo
Descripción: Para realizar cualquier tipo de cambio o modificación será necesario contar con una persona con conocimientos avanzados de HTML, CSS Y JS. El código es abierto está documentado y es completamente accesible para quien desee cambiarlo.

Identificador: RR02
Nombre: Internet
Prioridad: 3 tipos → Muy elevada, alta, media o baja
Fuente: Cliente o equipo de desarrollo
Necesidad: Primordial, opcional, baja necesidad
Verificabilidad: Alta, media o baja → Si se puede verificar que el requisito se cumple.
Estabilidad: si el requisito se modifica con el paso del tiempo
Descripción: La conexión a internet es fundamental para el uso de la aplicación.

Identificador: RR03

Nombre: Ordenador
Prioridad: 3 tipos → Muy elevada, alta, media o baja
Fuente: Cliente o equipo de desarrollo
Necesidad: Primordial, opcional, baja necesidad
Verificabilidad: Alta, media o baja → Si se puede verificar que el requisito se cumple.
Estabilidad: si el requisito se modifica con el paso del tiempo
Descripción: Es algo obvio pero no deja de ser un requisito que el dispositivo desde el que se acceda y se ejecuta la aplicación sea un ordenador con un SO. No incluye Chromebook ni tablets/smartphones.

Identificador: RR04
Nombre: Pantalla
Prioridad: 3 tipos → Muy elevada, alta, media o baja
Fuente: Cliente o equipo de desarrollo
Necesidad: Primordial, opcional, baja necesidad
Verificabilidad: Alta, media o baja → Si se puede verificar que el requisito se cumple.
Estabilidad: si el requisito se modifica con el paso del tiempo
Descripción: Para poder gozar de un buen funcionamiento recomendamos que la pantalla del ordenador desde la que se ejecute la aplicación tenga al mesno 13 pulgadas

- Requisitos de seguridad:

Identificador: RS01
Nombre: Login
Prioridad: 3 tipos → Muy elevada, alta, media o baja
Fuente: Cliente o equipo de desarrollo
Necesidad: Primordial, opcional, baja necesidad

Verificabilidad: Alta, media o baja → Si se puede verificar que el requisito se cumple.
Estabilidad: si el requisito se modifica con el paso del tiempo
Descripción: Para poder acceder a cualquier aplicación el usuario debe logearse previamente. Cada aplicación externa gozará de sus propias medidas de seguridad.

Identificador: RS02
Nombre: Username
Prioridad: 3 tipos → Muy elevada, alta, media o baja
Fuente: Cliente o equipo de desarrollo
Necesidad: Primordial, opcional, baja necesidad
Verificabilidad: Alta, media o baja → Si se puede verificar que el requisito se cumple.
Estabilidad: si el requisito se modifica con el paso del tiempo
Descripción: Cada usuario es diferente en cada aplicación externa. Nuestra aplicación no se encargará de autenticar de forma conjunta a un mismo usuario en los diferentes sistemas.

Identificador: RS03
Nombre: Password
Prioridad: 3 tipos → Muy elevada, alta, media o baja
Fuente: Cliente o equipo de desarrollo
Necesidad: Primordial, opcional, baja necesidad
Verificabilidad: Alta, media o baja → Si se puede verificar que el requisito se cumple.
Estabilidad: si el requisito se modifica con el paso del tiempo
Descripción: Cada usuario es diferente en cada aplicación externa, tal y como hemos mencionado en el requisito anterior. Por ello cada vez que un usuario desee cambiar la aplicación podrá tener acceso a hacerlo desde la propia aplicación pero encargándose la apliación externa

Identificador: RS04
Nombre: Sistema gestor de bases de datos
Prioridad: 3 tipos → Muy elevada, alta, media o baja

Fuente: Cliente o equipo de desarrollo
Necesidad: Primordial, opcional, baja necesidad
Verificabilidad: Alta, media o baja → Si se puede verificar que el requisito se cumple.
Estabilidad: si el requisito se modifica con el paso del tiempo
Descripción: Una de las razones principales y por las cuales nuestro sistema goza de total seguridad es debido a que trabaja como aplicación integradora. Cada aplicación externa goza de su propia seguridad.

- Requisitos de calidad:

Identificador: RC001
Nombre: Contraste
Prioridad: 3 tipos → Muy elevada, alta, media o baja
Fuente: Cliente o equipo de desarrollo
Necesidad: Primordial, opcional, baja necesidad
Verificabilidad: Alta, media o baja → Si se puede verificar que el requisito se cumple.
Estabilidad: si el requisito se modifica con el paso del tiempo
Descripción: La funcionalidad de cada uno de los requisitos se debe revisar y plantear según la etapa del proyecto en la que nos encontremos. **Hay que contrastar siempre que se cumple.**

Identificador: RC002
Nombre: Metrica empleada
Prioridad: 3 tipos → Muy elevada, alta, media o baja
Fuente: Cliente o equipo de desarrollo
Necesidad: Primordial, opcional, baja necesidad
Verificabilidad: Alta, media o baja → Si se puede verificar que el requisito se cumple.
Estabilidad: si el requisito se modifica con el paso del tiempo

Descripción: Durante toda la documentación del proyecto se llevará a cabo la metodología de métrica 3 .

Identificador: RC002
Nombre: Pruebas
Prioridad: 3 tipos → Muy elevada, alta, media o baja
Fuente: Cliente o equipo de desarrollo
Necesidad: Primordial, opcional, baja necesidad
Verificabilidad: Alta, media o baja → Si se puede verificar que el requisito se cumple.
Estabilidad: si el requisito se modifica con el paso del tiempo
Descripción: Durante el proyecto se realizarán diferentes pruebas y test para comprobar que la aplicación se encuentra lo suficientemente bien desarrollada como para su correcto funcionamiento y uso por parte del usuario final.

6.7 Especificación de los casos de uso.

A continuación (en este apartado del documento) llevaremos a cabo la especificación de los casos de uso.

Los casos de uso describen la interacción entre el sistema y los usuarios externos que conduce a la consecución de objetivos y acciones en concreto. Cada caso de uso incluye tres elementos principales, que son los siguientes:

- Actores. Son los usuarios externos al sistema que interactúan con el sistema.
- Sistema. El sistema se describe mediante requisitos funcionales que definen un comportamiento previsto del producto.
- Goles. Los propósitos de la interacción entre los usuarios y el sistema se esbozan como objetivos.

Existen dos formatos para representar casos de uso:

- **Especificación de casos de uso** estructurados en formato textual
- **Diagrama de casos de uso**

Una especificación de caso de uso representa la secuencia de eventos junto con otra información relacionada con este caso de uso. Una plantilla de especificación de caso de uso típico incluye la siguiente información:

1. Identificador: Permite a cada caso de uso identificarlo de manera univoca
2. Nombre: Campo significativo para el caso de uso
3. Actores: Indica que actores están relacionados con el caso de uso
4. Descripción: Explicación del caso de uso en concreto
5. Condición previa: funciones necesarias para que pueda darse el caso de uso
6. Pos condición posterior a la interacción: Indica el estado del sistema una vez ejecutado el caso de uso en cuestión.

Ejemplo:

Identificador:
Nombre: Nombre de la especificación de caso de uso
Actores del caso de uso: Usuario CRM, Usuario ERP, Usuario funcional, analista.
Descripción
Pre-condición o condición previa.
Pos condición posterior a la interacción

A diferencia de ello un diagrama de caso de uso no contiene muchos más detalles. Muestra una visión general de alto nivel de las relaciones entre los actores, los diferentes casos de uso y el sistema. Un diagrama de caso de uso incluye los siguientes elementos principales:

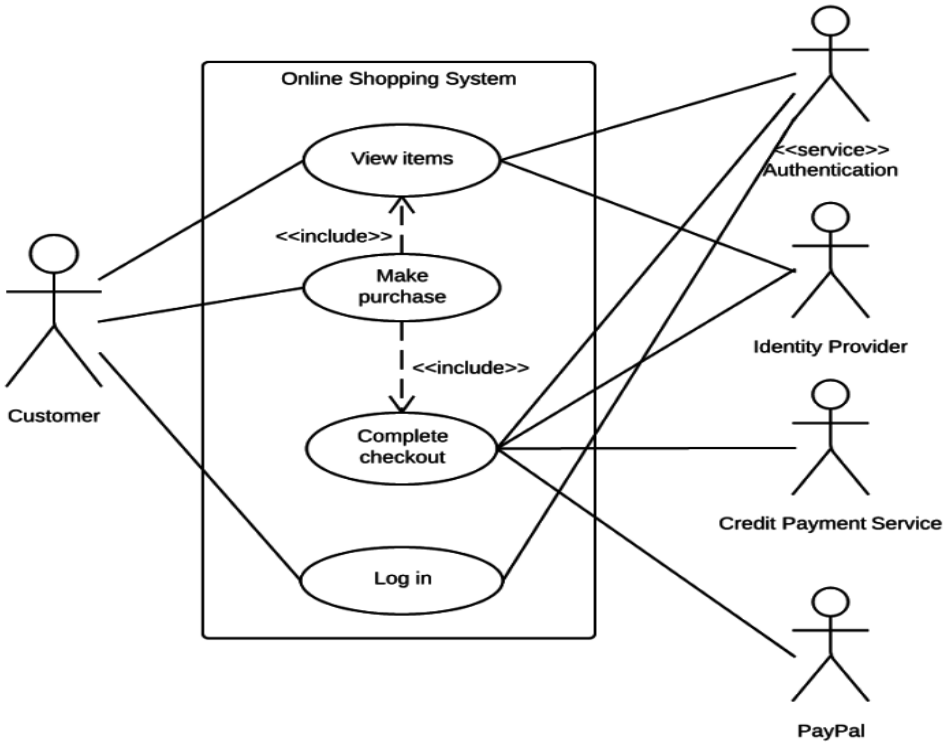
- **Casos de uso.** Usualmente dibujados con óvalos, los casos de uso representan diferentes escenarios de uso que los actores pueden tener con el sistema (iniciar sesión, hacer una compra, ver artículos, etc.).

- **Límites del sistema.** Los límites están delimitados por el cuadro que agrupa varios casos de uso en un sistema.

- **Actores**. Estas son las cifras que representan a los usuarios externos (personas o sistemas) que interactúan con el sistema.

- **Asociaciones**. Las asociaciones se dibujan con líneas que muestran diferentes tipos de relaciones entre actores y casos de uso.

Ejemplo de un diagrama de caso de uso:



- Especificación de los casos de uso:

En el caso de nuestra aplicación muchos de los casos de uso se corresponden con los de aplicaciones de terceros. A continuación, mostramos gran parte de ellos o los que hemos

considerado fundamentales. La mayoría son obvios y ya están definidos/hablados con el cliente.

Identificador: CU - 01
Nombre: Acceso a la aplicación de Salesforce
Actores del caso de uso: Usuario CRM
Descripción: El usuario entra en la aplicación CRM de Salesforce donde puede ver todas sus cuentas, oportunidades y leads y acceder a ellos.
Pre-condición o condición previa: Es necesario que el usuario acceda previamente con su licencia de Salesforce y haga Login con sus credenciales.
Pos condición posterior a la interacción: Los datos del usuario quedarán almacenados en la base de datos del CRM y de esta forma el sistema o la instancia de Salesforce en concreto mostrará toda su información.

Identificador: CU - 02
Nombre: Acceso a la aplicación de Gmail
Actores del caso de uso: Usuario funcional
Descripción: El usuario entra en la aplicación Gmail de Google donde puede ver todos sus correos, responderlos o acceder a ellos.
Pre-condición o condición previa: Es necesario que el usuario acceda previamente con su usuario de Google y haga Login con sus credenciales.
Pos condición posterior a la interacción: Los datos del usuario quedarán almacenados en la base de datos de Google y de esta forma el sistema o la instancia de Gmail en concreto mostrará toda su información de la cuenta

Identificador: CU - 03
Nombre: Acceso a la aplicación de Holded
Actores del caso de uso: Usuario ERP
Descripción: El usuario entra en la aplicación Holded donde puede ver toda su información de ERP
Pre-condición o condición previa: Es necesario que el usuario acceda previamente con su usuario de Holded y haga Login con sus credenciales.

Pos condición posterior a la interacción: Los datos del usuario quedarán almacenados en la base de datos de Holded y de esta forma el sistema o la instancia en concreto mostrará toda su información de la cuenta, o datos de su empresa tales como la facturación etc…

Identificador: CU - 04
Nombre: Acceso a la aplicación de Telegram
Actores del caso de uso: Usuario Analista/ comunicación /todos
Descripción: El usuario entra en la aplicación Telegram donde puede contactar con el resto de usuarios del sistema mediante mensajería instantánea.
Pre-condición o condición previa: Es necesario que el usuario acceda previamente con su información de móvil y escanee el código QR.
Pos condición posterior a la interacción: Los datos del usuario quedarán almacenados en la base de datos de Telegram de forma que se permitirá al usuario empezar a chatear instantáneamente con cualquier otro sistema

Identificador: CU - 05
Nombre: Acceso a la aplicación de Horbito
Actores del caso de uso: Usuario Analista/ comunicación /todos
Descripción: El usuario entra en la aplicación Horbito donde puede trabajar desde su escritorio remoto virtual
Pre-condición o condición previa: Es necesario que el usuario acceda previamente con su información personal como móvil/ contraseña
Pos condición posterior a la interacción: Los datos del usuario quedarán almacenados en la base de datos de Horbito de forma que se permitirá al usuario empezar a trabajar instantáneamente con el sistema.
Identificador: CU - 01
Nombre: Acceso a la consola de Salesforce
Actores del caso de uso: Usuario CRM
Descripción: El usuario entra en la aplicación CRM de Salesforce donde puede ver todas sus cuentas, oportunidades y leads y acceder a ellos, se dirije a la parte del Setup y desde ahí accede a la consola de programación de Salesforce

Pre-condición o condición previa: Es necesario que el usuario acceda previamente con su licencia de Salesforce y haga Login con sus credenciales.
Pos condición posterior a la interacción: El usuario debe tener permisos de codificación y acceso a la consola dentro de la herramienta para poder programar aplicaciones dentro de Salesforce

Identificador: CU - 03
Nombre: Acceso al gestor documental de Holded
Actores del caso de uso: Usuario ERP
Descripción: El usuario entra en la aplicación Holded donde puede gestionar toda la facturación de su empresa
Pre-condición o condición previa: Es necesario que el usuario acceda previamente con su información y tenga una licencia preprogramada para poder acceder
Pos condición posterior a la interacción: Cualquier gestión documetnal que realice el usuario ERP en el centro de facturación de la aplicación se quedará guardado.

Integrando plataformas como Salesforce CRM o Holded, nuestra aplicación dispone de una infinidad de casos de uso que se pueden encontrar según los requerimientos que tenga la empresa. Podrían ser miles las posibilidades y diferentes acciones de las que el usuario dispone gracias a nuestra aplicación y es por ello que solamente nos hemos centrado en el acceso a estas terceras partes desde nuestra aplicación en los casos de uso.

Entendemos que una vez iniciada sesión o aceptado el acceso a una plataforma externa desde nuestra aplicación será posible realizar infinidad de tareas relacionadas con el software empresarial, conllevando desde analítica de datos, hasta procesamiento de inteligencia artificial.

Es una de las principales ventajas y de los objetivos que buscábamos, trabajar como conector entre los principales softwares en la nube de la industria nos permite ofrecer infinidad de casos de uso.

- Diagramas de secuencia:

Gracias a los diagramas de secuencia podemos mostrar como los usuarios interactúan con la aplicación que hemos desarrollado. Tal y como veremos a continuación no solamente veremos como lo hacen con la nuestra, sino también con los accesos otorgados a las aplicaciones secundarias que ofrecemos.

Debido a que son miles los diagramas de secuencia que se podrían mostrar tan solo mostraremos unos ejemplos.

1. Acceso a la consola de Salesforce:

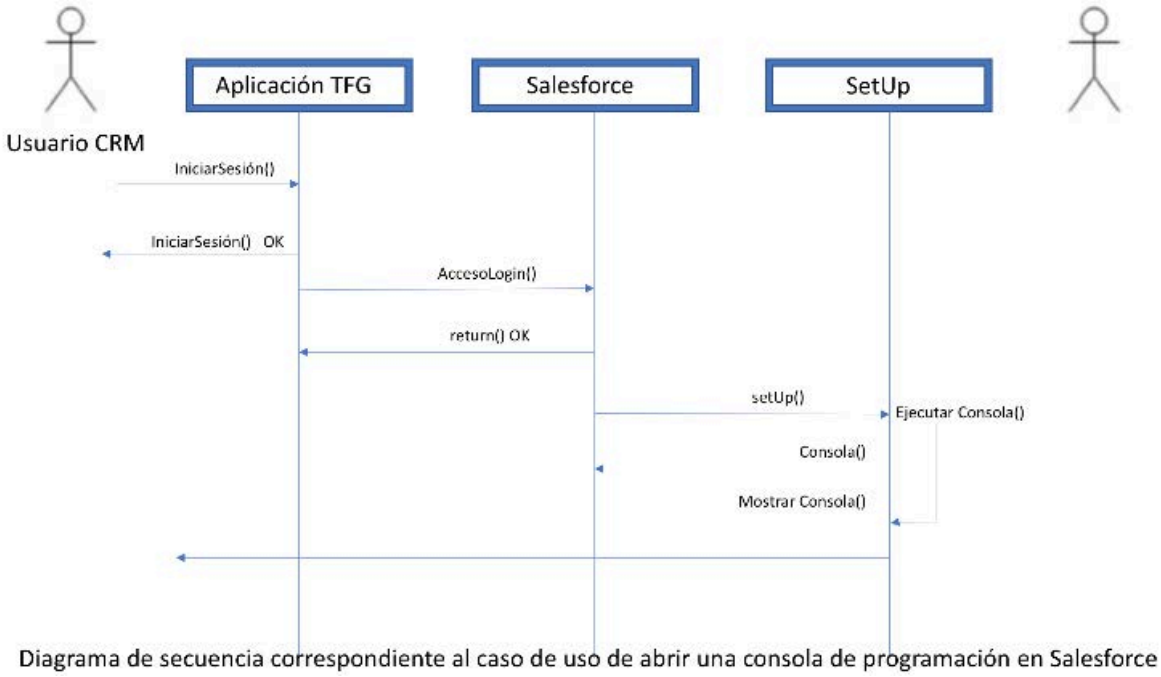


Diagrama de secuencia correspondiente al caso de uso de abrir una consola de programación en Salesforce

2. Acceso a la redacción de correos de Gmail:

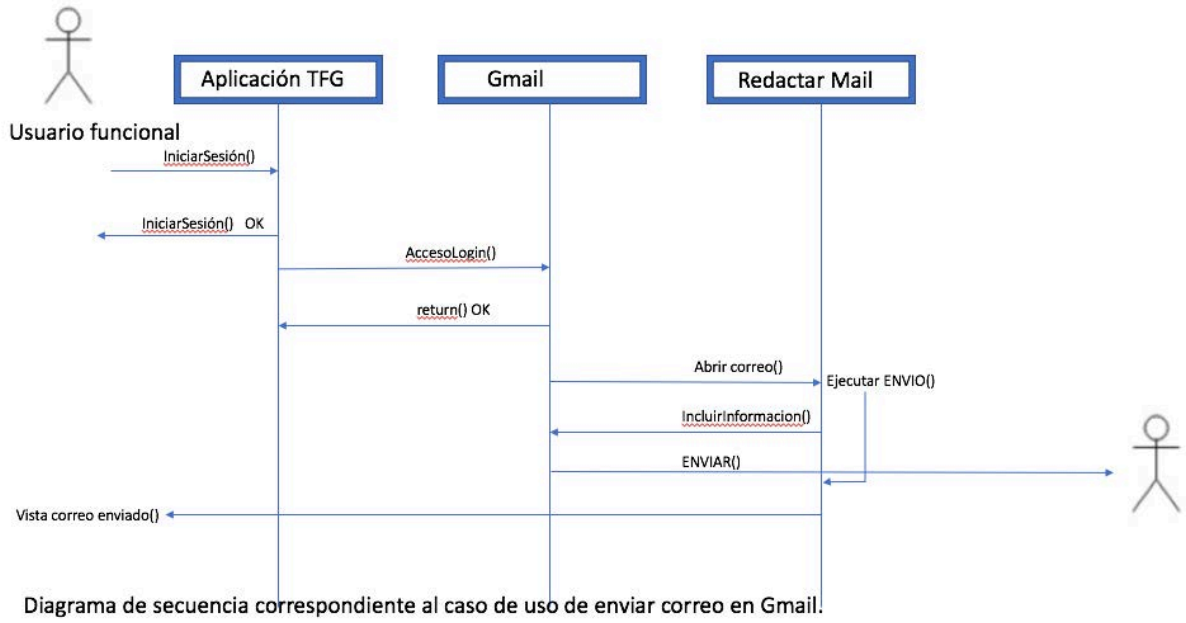


Diagrama de secuencia correspondiente al caso de uso de enviar correo en Gmail.

3. Acceso a cualquier aplicación de Horbito:

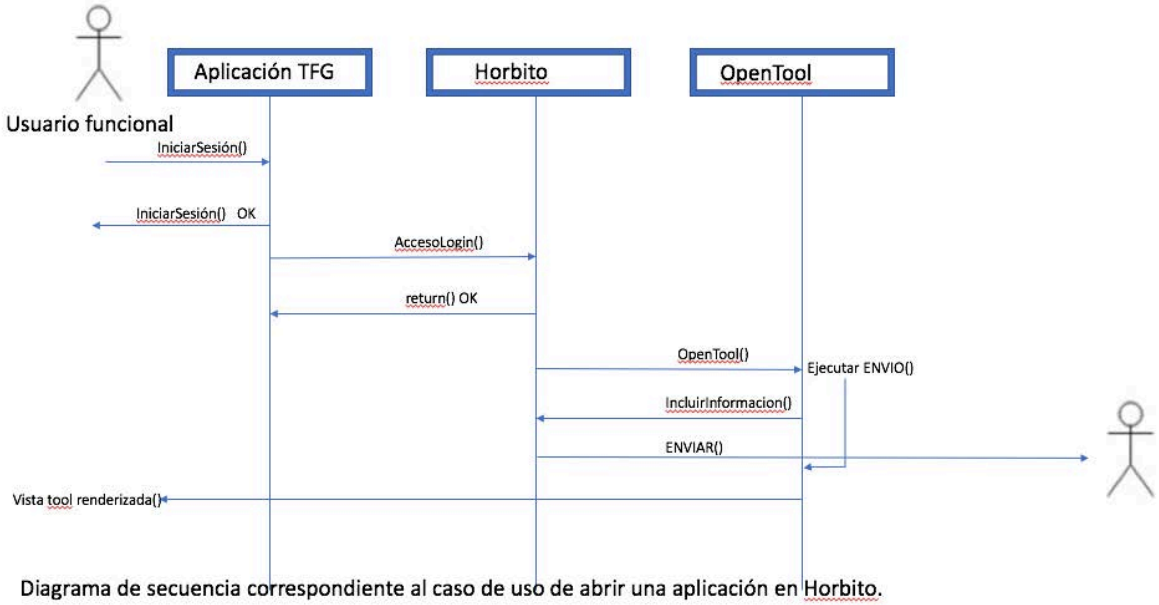


Diagrama de secuencia correspondiente al caso de uso de abrir una aplicación en Horbito.

4. Acceso al gestor documental de Holded:

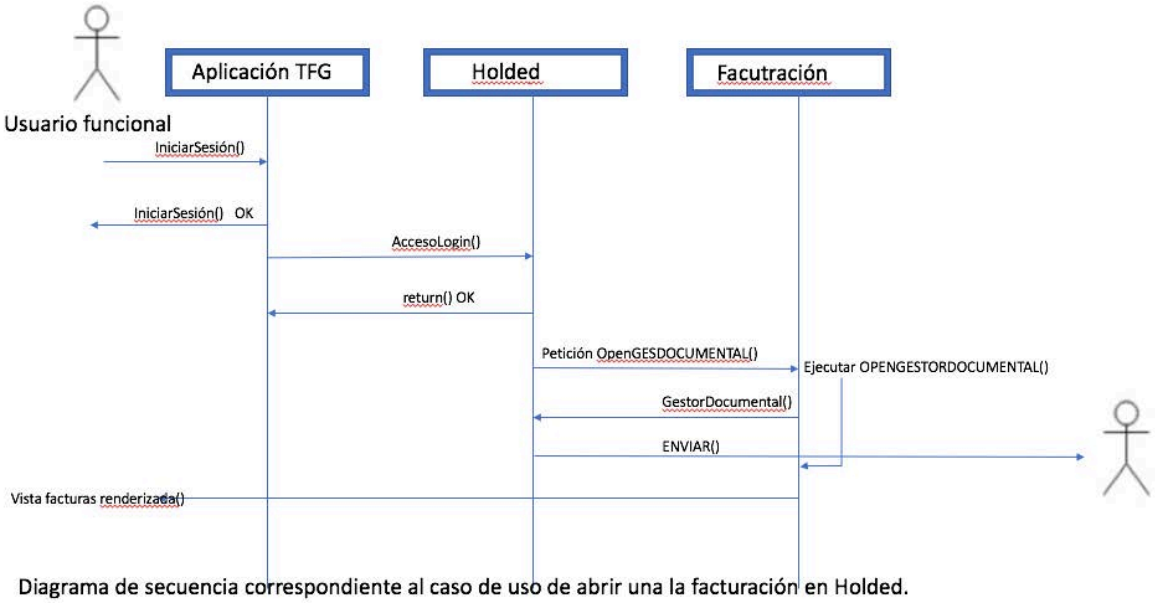


Diagrama de secuencia correspondiente al caso de uso de abrir una la facturación en Holded.

Tal y como podemos ver en los diagramas de nuestros casos de uso nuestra aplicación siempre funciona de la misma manera. Simplemente nos encargamos de la parte conectora que permite que el usuario interactúe con varios sistemas.

6.8 Análisis de la funcionalidad.

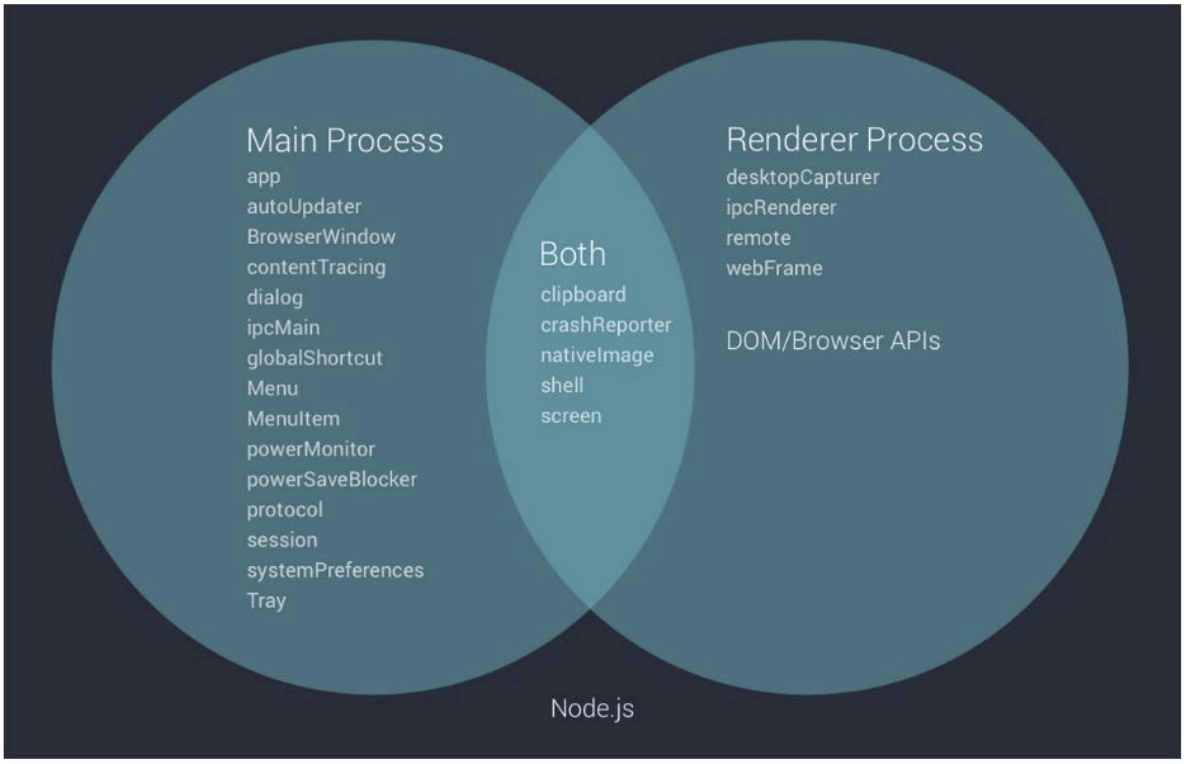
En cuanto a este apartado se describirá la estructura de programación en la que se basa nuestra aplicación. Tratándose de un conector hemos facilitado muchísimo la vida del usuario ofreciéndole un conjunto extenso de aplicaciones, pero reduciendo al máximo los costes de nuestra aplicación.

Por ello en vez de decidirnos por explicar todas las clases objetos y métodos de los que dota nuestra aplicación hemos preferido explicar la base del framework en la que se desarrolla y como esta esta construido, ya que al fin y al cabo es el código **CORE** de nuestra aplicación.

En este apartado se explica el funcionamiento CORE de la base para construir nuestra aplicación. Las bases de nuestra aplicación están sustentadas en el Framework Electron ya comentado antes, y es por eso que este apartado es de vital importancia

Electron es una plataforma que combina el motor de renderizado de Chromium y el sistema de tiempo de ejecución y módulos de Node.js.

Nuestra aplicación por tanto hereda el modelo multiproceso de Chromium, la aplicación principal, en un proceso separado con su propio espacio de memoria. Para el equipo de desarrollo, esto significa que puede reiniciar equipos individuales que se estrellen o que tengan otros problemas sin afectar al resto de la aplicación, así como la protección frente a problemas con los controladores de la GPU a través de un proceso de GPU independiente.



Este diagrama de Venn muestra las APIs de Electrón disponibles en cada tipo de proceso. También podemos observar que las APIs de Node.js están disponibles globalmente, mientras que sólo las APIs de DOM/Browser están disponibles en un renderizador.

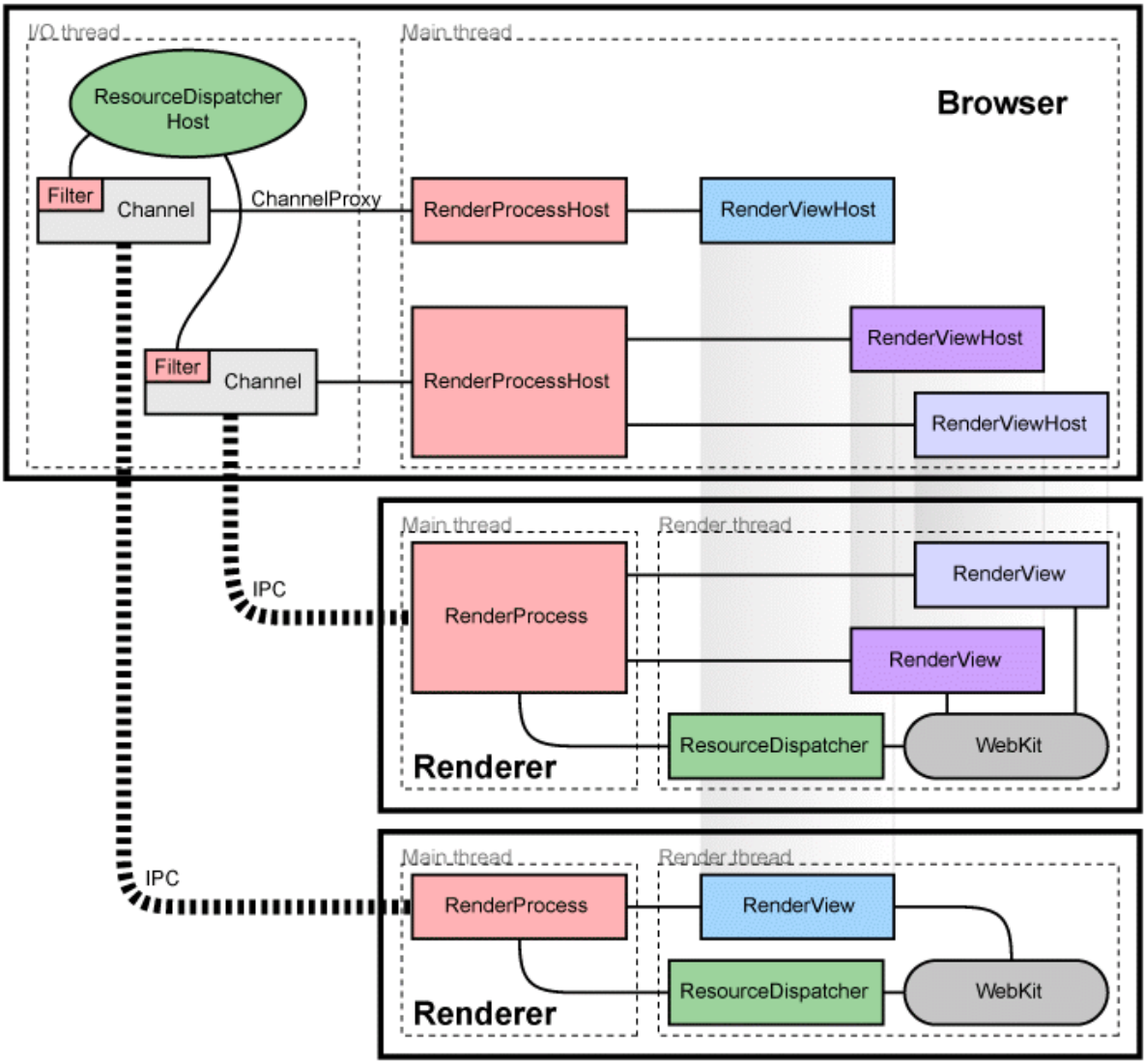
Hay varios enfoques para el multithreading en Electron. Los trabajadores Web son convenientes, pero carecen de la capacidad de usar módulos nativos. Forjar nuevos procesos

funciona como lo haría en Node, pero la falta de capacidad para usar la librería Electron obliga a usar IPC ((inter process comunication o más bien comunicación entre procesos), para tareas comunes y puede complicarse rápidamente. El uso de los procesos de renderizado como trabajadores garantiza toda la potencia de todas las herramientas de servidor de nodos disponibles como sustituto de la comunicación a través de IPC (inter process comunication), a la vez que se mantiene el acceso a los módulos y métodos nativos de la biblioteca de renderizado de Electron.

Como Electron empaqueta los archivos en un archivo ASAR de sólo lectura, no se puede incluir ningún archivo en el que tengamos que escribir (como una base de datos SQLite). En su lugar, estos pueden ser colocados en el directorio de Recursos donde permanecerán en la aplicación empaquetada.

Finalmente, hay que tener en cuenta el hecho de que en una aplicación empaquetada, algunas propiedades de los nodos no se comportan como el equipo de desarrollo esperaría. Y para mayor claridad, es necesario que hagan coincidir la estructura de archivos de la aplicación empaquetada con su código fuente.

En la imagen siguiente mostramos como funciona la conexión que existe entre procesos de NodeJs y Chromium.



La etiqueta WebView es un factor fundamental en el desarrollo de aplicaciones con este Framework y a continuación explicamos el porqué.

Aunque generalmente se confía en que la aplicación local se ejecute con acceso completo al escritorio y a Node.js, lo que permite que el contenido remoto acceda directamente a las funciones del escritorio y Node.js es inseguro - si alguien fuera a Man-In-The-Middle Slack, ¡tendría control total sobre las computadoras de los usuarios!
Para evitar esto, utilizamos una característica de Electron portado desde Chrome Apps llamada el elemento WebView (no relacionado con la vista WebView de Apple). Conceptualmente, este elemento HTML es similar a un iframe, ya que incluye otro sitio en línea como elemento de bloque. Sin embargo, en realidad crea un proceso de renderizado de

Chromium (navegador) separado y delega la renderización de contenido para su renderizador de alojamiento, de forma similar a como funciona el marco de trabajo del host del plugin de Flash.

Antes de que ocurra cualquier navegación, el equipo de desarrollo tiene la oportunidad de ejecutar código personalizado con la integración de Node.js habilitada, llamado "preload script". Este script se ejecuta antes de que se cree el DOM y antes de que la página tenga un origen, pero nos da acceso a las APIs de Electron y Node.js.

Una cosa que se puede hacer en nuestro script de precarga es establecer una clave en el objeto ventana. Esto permite exponer una API al lado webapp de las aplicaciones externas. Dado que definimos esta API, podemos establecer una Frontera de Seguridad que sólo concede al webapp ciertos métodos.

Hay algunas cosas que el cliente debe hacer para que este enfoque sea seguro:

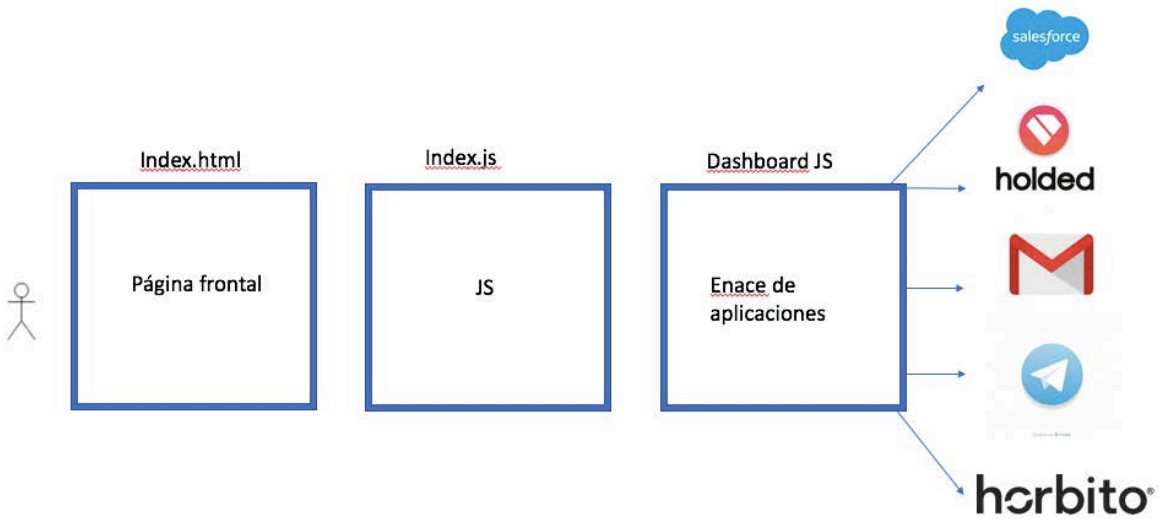
Debe asegurarse de no filtrar módulos de Node.js en la superficie de su API.

Debe tener cuidado con sus APIs, especialmente las que tienen que ver con las rutas de los archivos. Es importante asegurarse de que una llamada maliciosa de su API no pueda acceder a los datos del sistema de archivos de un usuario.

El equipo de desarrollo solo tiene que preocuparse por el acceso a los objetos JS a través de JavaScript, ya que poder ver los objetos Node.js a través de la pestaña de la consola de DevTools es, por lo general, seguro. DevTools tiene acceso a métodos V8 ocultos que JavaScript no tiene, por lo que el poder llegar a los objetos Node.js a través de, por ejemplo, la pseudovariable de "cierre" no es una preocupación.

Una vez entendida esta funcionalidad del framework podemos observar la capa superior en cuanto a nivel de desarrollo de nuestra aplicación.

CAPA SUPERIOR DE DESARROLLO DE LA APLICACIÓN – Estructura.



Index.html

Es sin duda una de las clases principals de nuestra aplicación. En ella definimos el tipo de navegación que temenos en el menu. Cada enlace a la plataforma web que queramos utilizar.

index.html

index.html ×

```
<!-- Sidebar -->
<div id="sidebar-wrapper">
  <ul class="sidebar">
    <li class="sidebar-main">
      <a ng-click="toggleSidebar()">
        Dashboard
        <span class="menu-icon glyphicon glyphicon-transfer"></span>
      </a>
    </li>
    <li class="sidebar-title"><span>Menu</span></li>
    <li class="sidebar-list">
      <a href="#">Aplicación TFG <span class="menu-icon fa fa-tachometer"></span></a>
    </li>
    <li class="sidebar-list">
      <a href="#/tables">Gmail <span class="menu-icon fa fa-table"></span></a>
    </li>
    <li class="sidebar-list">
      <a href="#/mail">Salesforce <span class="menu-icon fa fa-envelope"></span></a>
    </li>
    <li class="sidebar-list">
      <a href="#/github">Horbito <span class="menu-icon fa fa-cloud"></span></a>
    </li>
    <li class="sidebar-list">
      <a href="#/github">Dropbox <span class="menu-icon fa fa-mail"></span></a>
    </li>
    <li class="sidebar-list">
      <a href="#/github">Asana <span class="menu-icon fa fa-block"></span></a>
    </li>
    <li class="sidebar-list">
      <a href="#/github">Holded <span class="menu-icon fa fa-rule"></span></a>
    </li>
  </ul>
  <div class="sidebar-footer">
```

Index.js

Dashboard.js

6.9 Especificaciones generales de la interfaz de la plataforma.

A continuación, en este apartado mostraremos la manera en la que se van a comunicar los diferentes empleados de las empresas (es nuestro target de usuarios) con el sistema que se pretende desarrollar. El objetivo es comprobar y realizar un estudio exhaustivo sobre como se comporta el usuario con la plataforma para poder mejorar al máximo la interfaz haciéndola mas simple e intuitiva para su uso.

Es importante que a la hora desarrollar la interfaz del Sistema esta cumpla también con todos los requisitos de la misma.

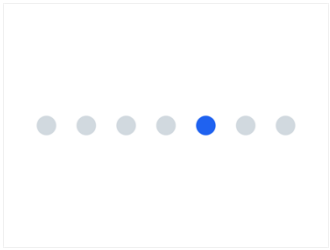
La interacción del usuario se ha convertido en un aspecto crucial de la construcción de un sitio web o una aplicación. Además de la simplificación, el contenido, la humanización y la personalización, se ha tenido en todo momento en cuenta que la integración también desempeña un papel importante para garantizar que el usuario encuentre fácil de comprender y navegar por el sitio web o la aplicación.

La psicología juega un papel importante en la experiencia del usuario con una aplicación. Entendiendo cómo se perciben nuestros diseños, podemos hacer ajustes para que las aplicaciones que creamos sean más efectivas en el logro de los objetivos del usuario.

Para ayudar a entender la percepción del usuario, vamos a introducir algunos principios de diseño que creemos que son los más importantes, y también proporcionar ejemplos comunes de estos principios en la práctica. Empecemos:

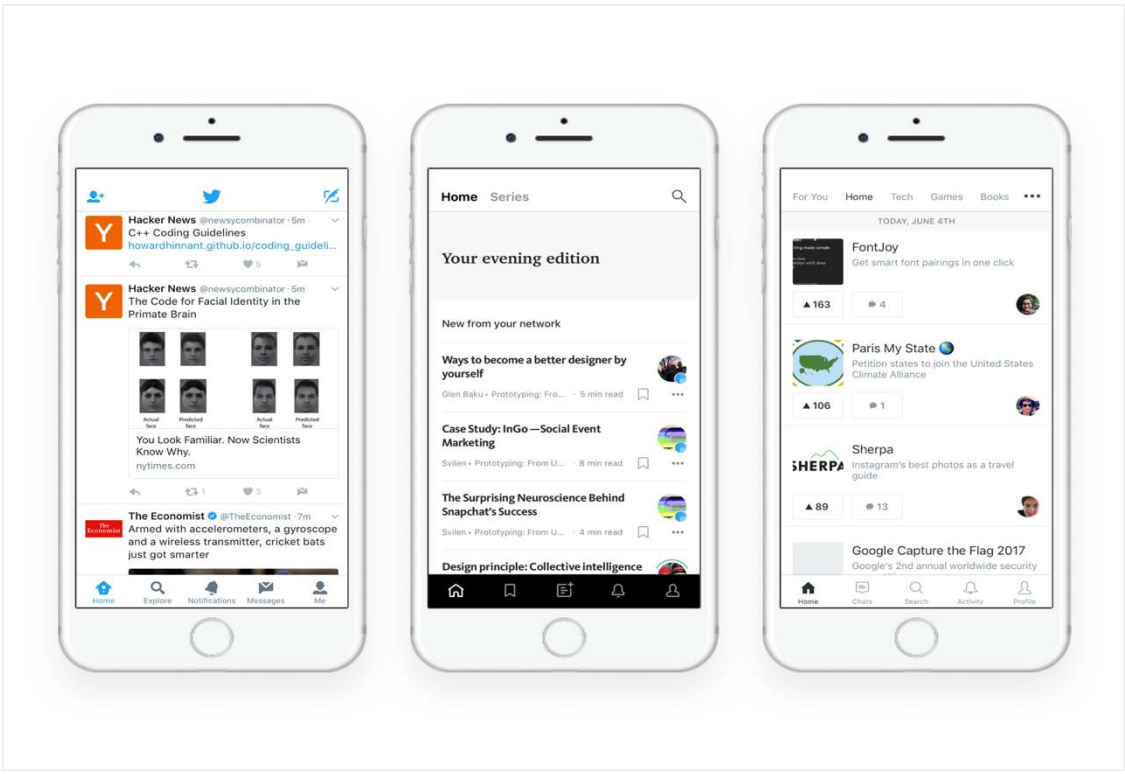
1. ***El efecto Von Restorff:***

El efecto Von Restorff (también conocido como efecto de aislamiento) predice que cuando hay varios objetos similares presentes, ¡es más probable que se recuerde el que difiere del resto. Esta es la razón principal por la que todas las llamadas a la acción (CTAs) se ven diferentes del resto de los botones de acción en un sitio o aplicación



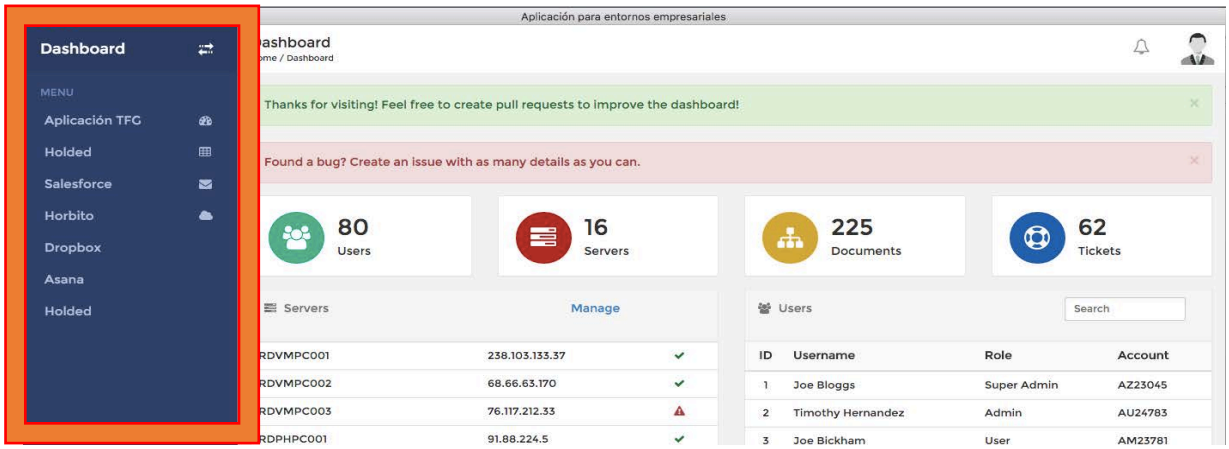
2. Efecto de posición en serie

El efecto de posición en serie es la propensión de un usuario a recordar mejor el primer y último elemento de una serie.



De izquierda a derecha tenemos las aplicaciones Twitter, Medium, ProductHunt.
Esta es la razón por la que la mayoría de las aplicaciones hoy en día abandonan el menú clásico y optan por una navegación de barra inferior o superior, colocando las acciones de usuario más importantes a la derecha o a la izquierda. En la imagen de arriba, puedes ver algunos ejemplos de aplicaciones populares de iOS. Cada uno de ellos coloca los elementos "Home" y "Profile" a la izquierda y a la derecha, teniendo en cuenta el efecto de posición en serie. En este caso, tratándose de una aplicación empresarial se ha decidido realizar un diseño de menú tradicional pero no se subestima la idea de desarrollar una navegación entre pestañas de este modo en un futuro.

Ejemplo de menú selector de aplicaciones en nuestra aplicación:



3. La carga cognitiva

Se refiere a la cantidad total de esfuerzo mental que se utiliza en la memoria de trabajo de una persona. En pocas palabras, es la cantidad de pensamiento que necesitas ejercitar para completar una tarea específica.

"La carga cognitiva es la cantidad de pensamiento que necesitas ejercitar para completar una tarea específica."

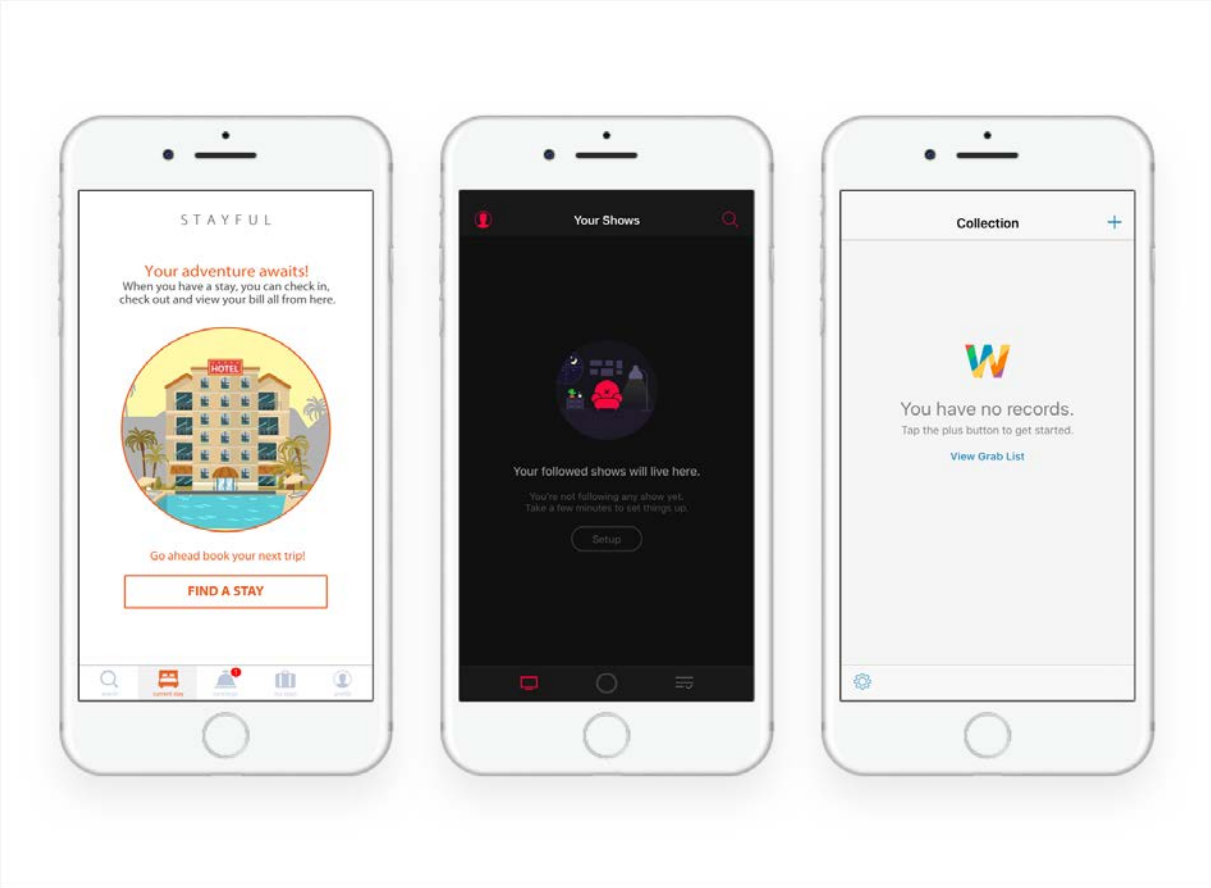
La teoría de la carga cognitiva se puede diferenciar en tres tipos:

- Carga cognitiva intrínseca
- Carga cognitiva extraña
- Carga cognitiva alemana

Nos referiremos a los tipos Intrinsic y Germane ya que se han considerado que son los más aplicables al diseño de UX.

La carga cognitiva intrínseca es la dificultad asociada con un tema didáctico específico. Es la razón principal por la que la micro-copia y la copia juegan un papel importante en una buena experiencia de usuario.

Por ejemplo, la mayoría de las veces en los estados vacíos de las aplicaciones, pedimos a los usuarios que completen una tarea. Aquí, la copia debe ser corta, sencilla y con las palabras adecuadas para que el usuario pueda seguir fácilmente las instrucciones. Mostramos un ejemplo.



4. Carga Cognitiva Alemana

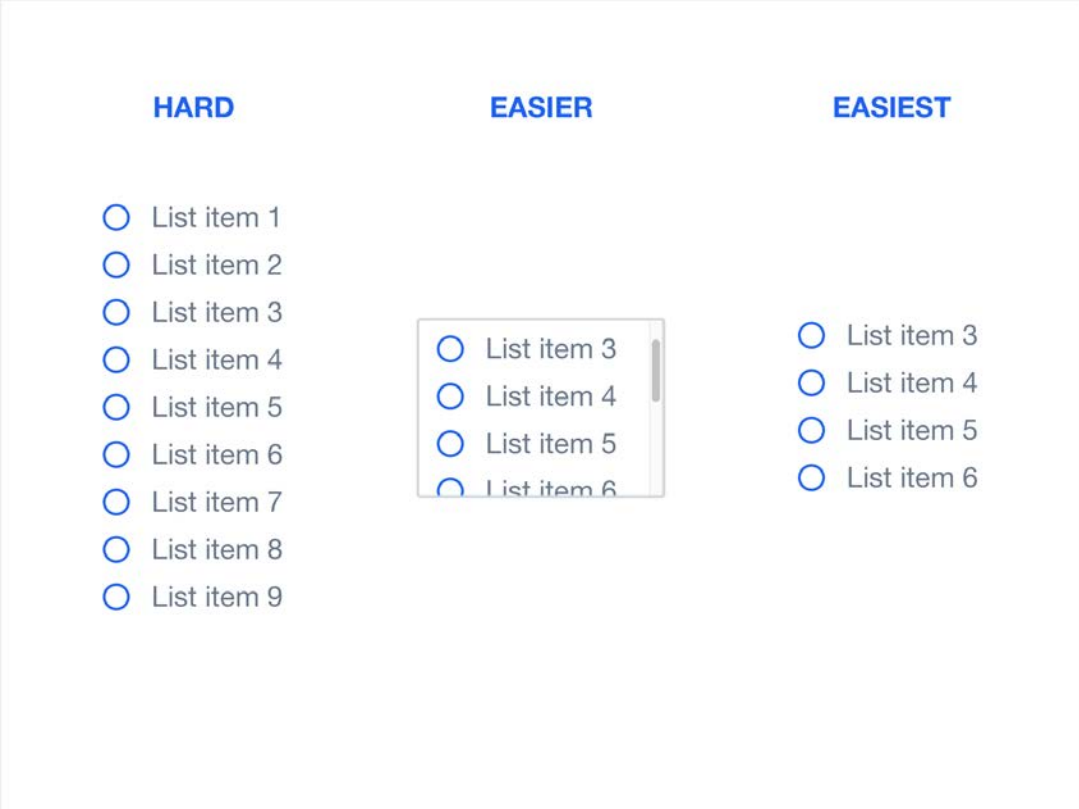
La carga cognitiva alemana es la carga cognitiva dedicada al procesamiento de la información y a la construcción de esquemas. Los esquemas describen un patrón de pensamiento que organiza categorías de información y cualquier relación entre ellas.

Una de las razones por las que se usan patrones de diseño es porque son algo para lo que estamos programados por defecto, así que es más fácil para los usuarios reconocer y aprender algo nuevo si pueden discernirlo en un patrón a partir de algo que ya entienden.

"Es más fácil para los usuarios aprender algo nuevo si pueden discernir un patrón a partir de algo que entienden."

Ley de Hick's

La Ley de Hick es el principio más popular, junto con las Leyes Gestalt.
También es muy sencillo de entender y practicar. Hick's Law describe que el tiempo que le toma a una persona tomar una decisión depende de las opciones que tiene a su disposición. Así que si el número de opciones aumenta, el tiempo para tomar una decisión aumenta logarítmicamente.
Un buen ejemplo de la Ley de Hick's que se aplica al diseño de experiencias de usuario son las listas:



Ley de Proximidad

La ley de proximidad es parte de las Leyes Gestalt de la Organización Perceptual, y establece que los objetos que están cerca, o próximos entre sí, tienden a ser agrupados. Para decirlo en términos más simples, nuestro cerebro puede asociar fácilmente objetos cercanos entre sí, mejor que los objetos que están muy separados. Esta agrupación ocurre porque los humanos tienen una tendencia natural a organizar y agrupar las cosas.

"La Ley de la Proximidad establece que los objetos que están cerca, o próximos entre sí, tienden a ser agrupados."

En el ejemplo anterior, hay 72 círculos. Reconocemos los círculos en grupos, basándonos en la distancia entre ellos. También percibimos que hay un grupo de 36 círculos en el lado izquierdo de la imagen, y 3 grupos de 12 círculos en el lado derecho de la imagen.

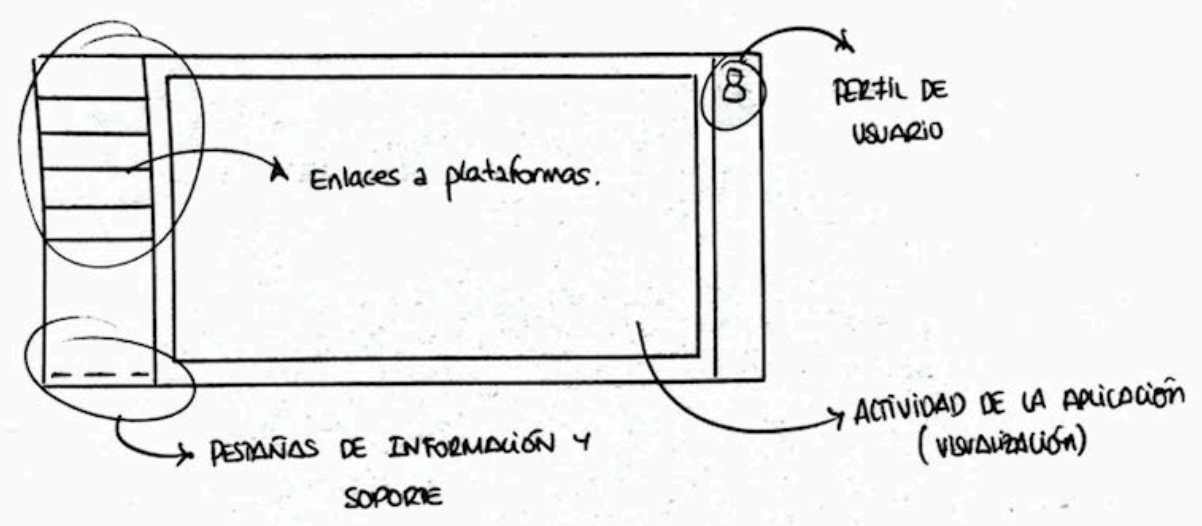
Creo que este ejemplo deja claro que es necesario agrupar las cosas cuando se diseña una interfaz de usuario, así como la importancia de ser cuidadoso a la hora de armar las cosas, ya que los usuarios pueden pensar naturalmente que están asociados entre sí.

Toda está información se ha extraído de un estudio detallado sobre el autor Thanasis Rigopoulos. El diseñador del equipo junto al equipo de desarrollo hizo mucho hincapié durante el proyecto en seleccionar tan solo aplicaciones que cumpliesen con este sistema.

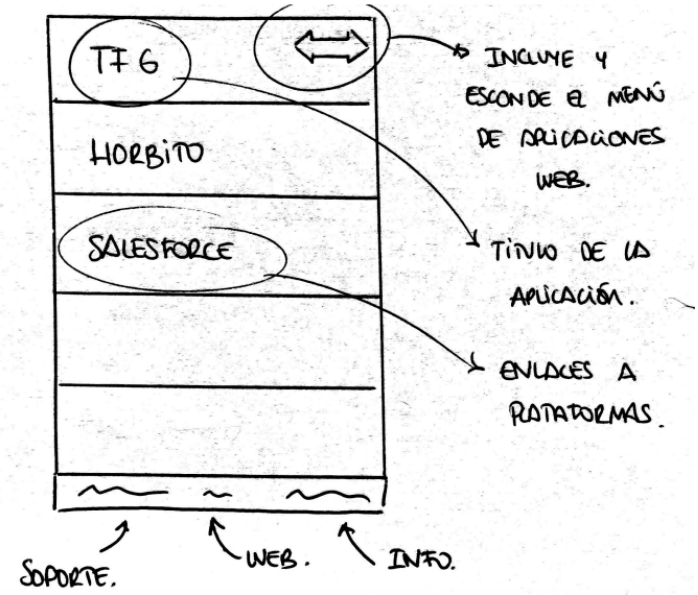
6.10 Prototipo de la interfaz.

A continuación, incluimos varios bocetos sobre el diseño de la interfaz a mano. Todos los patrones y métricas han sido evaluadas y seguidas por el diseñador de la aplicación.

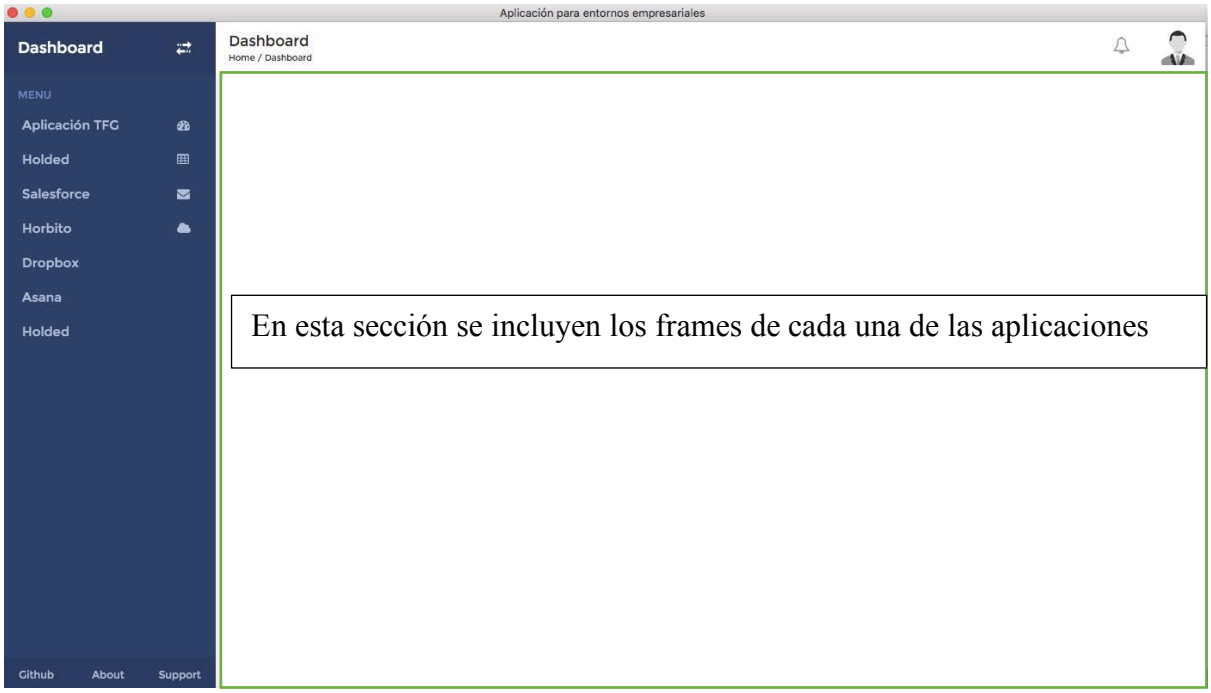
Diseño principal:



Menú de aplicaciones:



En la siguiente imagen podemos ver como ha quedado el diseño de la aplicación de escritorio.



Tal y como se puede observar tanto nuestra aplicación como todas las aplicaciones cloud que se incluyen con la nuestra poseen las propiedades básicas de diseño y UX necesarias para cumplir con los requisitos mínimos.

Capitulo 7. Diseño del sistema.

Introducción.

Debido a la complejidad de nuestra aplicación hemos decidido documentar tan solo una de las aplicaciones web que considerábamos interesantes. La aplicación web más compleja ha sido Salesforce debido a su modelo multitenant y la forma en la que la metadata actúa con los datos.
El modelo en que se estructura el sistema externo es vital para poder desarrollar el proyecto de forma posterior, por eso, para la realización de este documento se ha llevado a cabo un estudio detallado y exhaustivo de las diferentes plataformas web.

Alcance.

En todos los sistemas se ha llevado la misma metodología de análisis para comprobar que cumplían con los requisitos esperados por el equipo de desarrollo

- Definición de la arquitectura del sistema
- Normas seguidas de diseño
- Análisis de la funcionalidad
- Diseño de clases:
- Diseño físico de datos:
- Verificación y aceptación de la arquitectura del sistema:
- Plan de pruebas:

Definición de la arquitectura del sistema.

Salesforce ofrece una disposición central de las administraciones a cada uno de sus clientes en la nube multiusuario. Independientemente de la extensión del negocio, el usuario obtiene acceso a poder registrar, y almacenar información y características centrales similares.

En las Arquitecturas Basadas en Metadata, la “multitenancy” es viable justo cuando puede soportar aplicaciones que son confiables, adaptables, actualizables, seguras y rápidas. Es difícil hacer un ejecutable de aplicación estáticamente ordenado que pueda satisfacer estas y otras dificultades de “multitenencia”

Característicamente, una aplicación multiusuario debe ser de naturaleza dinámica, o polimórfica, para satisfacer los deseos individuales de los diferentes habitantes y sus clientes.

Por lo tanto, se han desarrollado planes de aplicación multiusuario para utilizar un motor en tiempo de ejecución que produce partes de la aplicación a partir de metadatos - información sobre la aplicación en sí misma. En una ingeniería basada en metadatos caracterizada, existe una división inconfundible del motor de tiempo de ejecución ordenado (kernel), la información de la aplicación, los metadatos que muestran la utilidad básica de una aplicación y los metadatos que se relacionan con la información y las personalizaciones de cada ocupante.

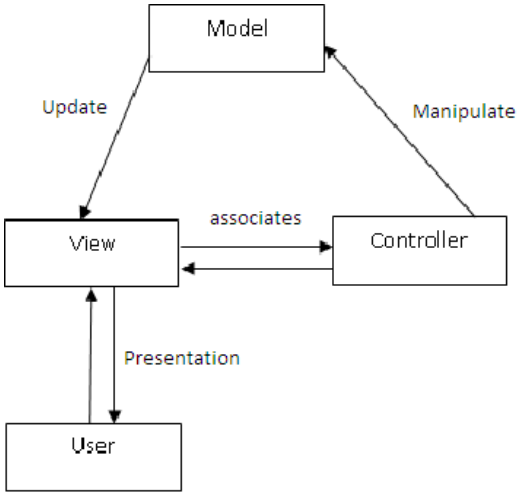
Por ello mismo, siendo uno de los lideres en la industria como proveedor de aplicaciones web para el mundo empresarial hemos elegido Salesforce para llevar a cabo un análisis sobre el diseño del sistema.

La arquitectura de Salesforce no es un resultado sorprendente de una serie aleatoria de experimentos de éxito y ensayo. Todas las características de su arquitectura han sido cuidadosamente planificadas y colocadas justo donde se requiere. Si nos fijamos y echamos un vistazo a su arquitectura se puede entender la mayor parte de su funcionalidad.

MVC es uno de los patrones de diseño de arquitectura de software más utilizados, que divide el diseño en tres componentes básicos: Modelo, Vista y Controlador. Salesforce proporciona a Visualforce una interfaz para desarrollar rápidamente aplicaciones siguiendo la arquitectura MVC sobre la nube.

En Visualforce puede implementar MVC utilizando tanto objetos estándar como personalizados. También puede implementar MVC utilizando 3 páginas de objetos, componentes y controladores de Salesforce recientemente introducidos.

Estas páginas funcionan como las páginas JSP o ASP y ofrecen la mejor presentación según las necesidades del usuario. Cada Vista tiene un controlador asociado. El desarrollador puede usar algún controlador estándar o puede escribir el suyo propio usando el lenguaje de programación Apex. VF también tiene controladores autogenerados para interactuar con las bases de datos.



MODELO

Los objetos de base de datos de Salesforce se conocen como Modelo de datos. Consiste en los objetos estándar de Salesforce como cuentas, contactos, oportunidades y clientes potenciales, etc. También soporta cualquier objeto personalizado creado según los requisitos del cliente. El tipo de esquema que se debe utilizar y el tipo de datos que se deben utilizar para representar se decide aquí. Podemos considerar el ejemplo de los objetos como modelo en la fuerza de ventas. Como cada entidad está mapeada a algunos sObjetos.

Campo, Objeto y Relaciones se encuentran en la sección Modelo de la arquitectura MVC.

VISTA

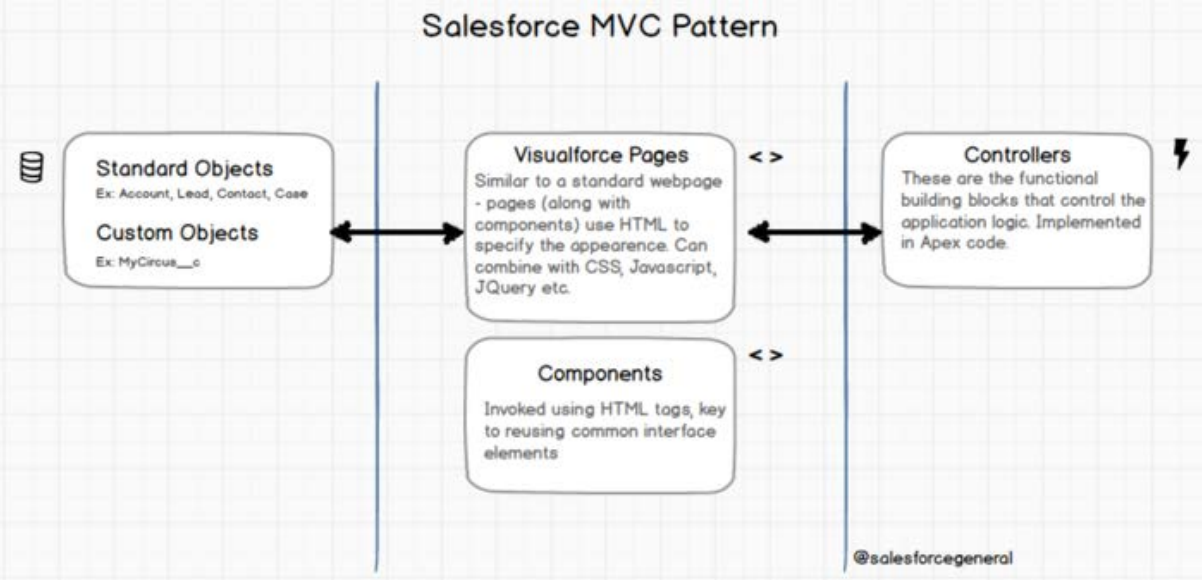
La presentación de la información al usuario no es más que la Vista. Esta es una interfaz de usuario dada al usuario para la interacción con el sistema. La vista consiste en páginas y componentes visuales de la fuerza de ventas. Las páginas pueden estar enlazadas con componentes de fuerza visual. Estas páginas utilizan HTML para preparar el diseño de la aplicación. Para referir cada página se enlaza con una URL única similar a la de las páginas web.

Componentes - El código reutilizable o bloque de código se denomina componente. Estos componentes se pueden estilizar con CSS y se reutilizan siempre que sea necesario.

Las páginas de fuerza visual, las pestañas y las clases de diseño de página se encuentran bajo la Vista.

CONTROLADOR

La lógica de negocio se implementa en el controlador. Estos son los bloques de construcción de la lógica real usando el lenguaje Apex. Las páginas de la vista interactúan con el controlador utilizando componentes. Salesforce dispone de un controlador de pre-construcción para algunas acciones estándar como guardar, editar, etc. Las clases Workflow, Triggers y Apex se encuentran bajo este nivel.



Cloud computing es el término en el que el consumidor puede obtener servicios como soporte o desarrollo de software, desarrollo de plataformas, gestión y compartición de infraestructuras, e-business (m-business) a través de Internet. El término nube se inspiró en el símbolo de la nube utilizado para representar a Internet en los diagramas de flujo.

La idea básica detrás de la computación en nube es que cualquier persona puede acceder a los servicios a través de Internet en cualquier momento y desde cualquier lugar. El servicio que proporciona el cloud computing puede tener características como elasticidad, dinamismo, rentabilidad, servicios bajo demanda, paga por lo que se utiliza.

Estas características tienen diferentes significados para los diferentes tipos de usuarios de la nube (stakeholders). Con el fin de garantizar un entendimiento común, los potenciales actores de la nube se caracterizan en cuatro roles generales:

- **Usuarios finales**: consumidores de los servicios de aplicaciones en la nube
- **Proveedores de servicios cloud (CSPs)** : Proveedor de capacidades cloud
- **Cloud Tool Providers (CTPs) :** Proveedor externo de herramientas de soporte y gestión de la nube.
- **Proveedores de Aplicaciones Cloud (CAVs):** Proveedor de servicios en la nube

El cloud computing es un modelo de prestación de servicios de TI a través de Internet y Salesforce es un gran ejemplo de ello. Los servicios se colocan en un lugar remoto (registro UDDI). Los recursos se recuperan de Internet a través de herramientas y aplicaciones basadas en la web. Estos servicios se dividen principalmente en tres categorías:

- Software-as-a-Service (Saas).
- Plataforma como servicio (Paas)
- Infraestructura como servicio (Iaas)

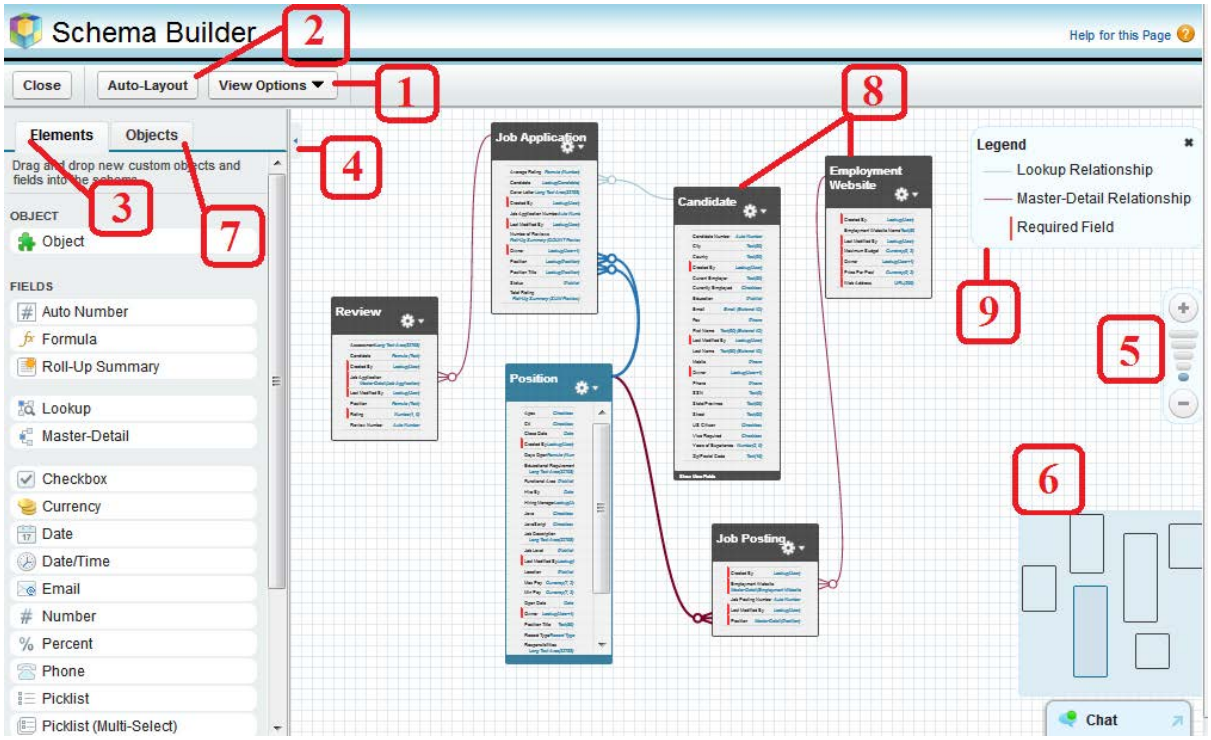
Nuestro objetivo es proporcionar al usuario acceso a varios servicios que ofrecen las mismas cualidades desde una sola aplicación de escritorio.

Diseño de clases.

En la plataforma de Salesforce el diseño de las clases dependerá de como decida tenerlo la empresa que compre nuestra aplicación. Una de las ventajas principales de la plataforma es que tiene una herramienta llamada schema builder en la que se muestra como están relacionados todos los objetos creados dentro de la plataforma.

Schema Builder es una herramienta de Salesforce.com que se utiliza para ver y gestionar objetos, campos y relaciones entre objetos en una interfaz gráfica. Está disponible en todas las ediciones de Salesforce.com. Schema Builder no es la única opción para ver y gestionar objetos, campos y relaciones entre objetos. Es tan grande el modelo de datos de la plataforma que nos ha sido imposible incluir todo el esquema.

A continuación mostramos un ejemplo como imagen



El esquema puede ser visto como un modelo, plano o representación gráfica. Definir un esquema en términos de tecnología de Salesforce es organizar los objetos, sus campos y relaciones. Podemos decir que una colección de objetos con nombre puede ser llamada como un esquema. Esta herramienta proporciona un marco estructurado para representar los objetos y su asociación.

Al igual que el esquema de base de datos, que se refiere a cómo se construye una base de datos, el esquema de Salesforce también proporciona un plano de cómo se organizan sus objetos. Por ello se utiliza como herramienta de diseño y modelado de la base de datos ER. De forma similar, salesforce proporciona una herramienta para manipular el esquema llamado Schema Builder. El generador de esquemas elimina la necesidad de hacer clic de página en página para encontrar los detalles de los objetos, campos y relaciones en su esquema.

Exploremos las características proporcionadas por el constructor de esquemas

1. Existen diferentes opciones de visualización disponibles:

- Mostrar nombres de elementos muestra los nombres del sistema o Mostrar etiquetas de elementos muestra valores de texto.
- Visualizar/suprimir relaciones de ordenación
- Visualizar/ocultar leyenda

2. Auto-Layout ajusta automáticamente la disposición de los objetos en la vista. Una vez que se hace el diseño automático, no se puede deshacer. Cerrar permite guardar el esquema y cerrar el generador de esquemas.

3. La ficha Elementos permite arrastrar y soltar nuevos objetos y campos personalizados en el esquema (incluidas las relaciones de ordenación, la fórmula, el resumen de rollup, etc.).

4. Permite colapsar la Flecha izquierda y expandir la Flecha derecha.

5. Permite acercar y alejar el zoom.

6. Da una forma minada de tu esquema. Puede arrastrar el rectángulo y ver la parte necesaria, y desplazarse por el esquema

7. La ficha Objetos permite seleccionar los objetos que se mostrarán en el lienzo. Puede buscar los objetos en el cuadro de texto Búsqueda rápida. Las opciones disponibles para la pestaña de objetos son las siguientes:-

- Todos los objetos
- Objetos seleccionados
- Objetos estándar
- Objetos personalizados
- Objetos del sistema

8. Estos son los objetos del lienzo. Aquí se puede realizar varias tareas. Si se hace clic en Símbolo, se puede:-

- Ocultar objeto sobre lienzo
- Editar propiedades de objeto
- Borrar objetos
- Ver objetos

- Ver diseños de página

Si se hace clic con el botón derecho del ratón en los campos de los objetos, se podrá:-

- Ver campos
- Editar propiedades de campo
- Administrar permisos de campo
- Borrar campos
- La leyenda indica las relaciones y los valores requeridos. Si ponemos el ratón sobre cualquier relación, se muestra el detalle.

Así que se puede concluir que Schema Builder proporciona una representación gráfica de los objetos de salesforce.com junto con sus detalles, como los campos necesarios, los valores de campo y la forma en que se relacionan los objetos, indicando las relaciones de búsqueda y de detalle maestro. Nos permite de este modo identificar fácilmente los campos requeridos, la búsqueda y la relación de detalles maestros mediante los diferentes códigos de color proporcionados por el organizador de la vista de esquema. También da la posibilidad de añadir objetos y campos directamente desde el generador de esquemas.

Verificación y aceptación de la arquitectura.

Como en cualquier proyecto de software se debe garantizar la calidad de las especificaciones y la estructura de la plataforma debe ser conforme a las medidas y requisitos definidos. Es por ello que es importante dejar claro que este apartado se debe considerar.

Las dos preguntas principales que siempre debe hacerse antes de considerar mostrarle a un cliente un producto son:

- ¿Está verificado?
- ¿Está validado?

Aunque ambas pueden parecer similares y ambas se refieren a lo que el cliente quiere construir, se refieren a dos cosas completamente diferentes, y nunca deben confundirse cuando se trata de asegurarse de que la aplicación esté lista para ser entregada.

Verificación

Este es el proceso de evaluación de código para asegurarse de que los productos de cualquier fase de desarrollo satisfacen las condiciones que se determinaron al principio de dicha fase. Esto significa que, aunque no evalúa el producto final, se asegura de que se va por el buen camino. Al verificar nuestro código, se decide si nuestro software satisface o no los requisitos especificados en dicha fase.

La pregunta primordial será: ¿Estamos construyendo bien el producto?
En este paso, siempre debe mantener en su cabeza la pregunta "¿estamos construyendo bien el producto? De esta manera, usted sabe que está comprobando si el programa funciona o no como se esperaba.

Validación

Esto se refiere al proceso de evaluación de nuestro código para asegurarnos de que satisface los requisitos comerciales de nuestro proyecto. Asegura que al final del proyecto, nuestro software hace exactamente lo que el cliente necesita, y esto está directamente relacionado con las especificaciones que se especificaron al principio del proyecto. Esto también significa que se está asumiendo que se tienen las especificaciones correctas desde el principio. Si no, todo este paso es inútil, ya que el software nunca será validado si no se comprueban todos los requisitos reales establecidos por el cliente.

¿Estamos construyendo el producto adecuado?
Hay dos formas principales de realizar la validación de software, pero ambas se centran en la pregunta "¿se está construyendo el producto adecuado?

Si la validación se realiza internamente, asumimos que los objetivos que nos fijamos desde el principio fueron interpretados correctamente, y si el proyecto cumple con las expectativas establecidas en los requisitos, decimos que está internamente validado.

Si la validación se realiza externamente, son el cliente, los socios y las partes interesadas quienes deciden si el producto satisface o no sus necesidades. Ya que ellos son los

responsables del proyecto, se podría decir que esta es la verdadera verificación, ya que ellos son los que realmente saben cómo quieren que se comporte su programa.

A continuación, mostramos los diferentes tipos de pruebas que se han llevado a cabo para poder validar el programa realizado en el proyecto.

Encontramos diferentes tipos de pruebas:

Pruebas unitarias: Se trata de que verifiquen que la funcionalidad de la aplicación sea la correcta.

Pruebas de integración: Se trata de que verifiquen que cada una de las plataformas externas a las que debemos conectarnos funcionen de la forma correcta.

Pruebas de aceptación: Tratan de validar la aplicación de escritorio llevada a cabo.

Pruebas de implantación: El desarrollo de esta parte de pruebas se llevará a cabo en la segunda versión de la aplicación.

Especificación técnica de las pruebas.

Cada prueba estará definida por la siguiente tabla en la que se encuentra la información necesaria recogida. La estructura de las tablas es la siguiente:

Identificador: TIPO DE PRUEBA
Nombre: Nombre de la prueba
Tipo: Tipo de prueba, unitaria, integración, aceptación, o implantación.
Descripción: El usuario entra en la aplicación CRM de Salesforce donde puede ver todas sus cuentas, oportunidades y leads y acceder a ellos.
Pre-condición: Qué requisitos es necesario que cumpla la prueba.
Instrucciones de prueba: Qué es necesario para que se cumpla la prueba
Criterios de aceptación: Criterios de aceptación de la prueba.

Pruebas unitarias.

Identificador: U001
Nombre: Acceso Gmail
Tipo: Tipo de prueba, unitaria
Descripción: El usuario entra en la aplicación Gmail donde puede ver todas sus correos, contactos listas etc...
Pre-condición: El usuario debe tener disponible el icono de la aplicación para abrirla.
Instrucciones de prueba: - **Abrir la aplicación mediante su icono en el escritorio** - **Acceder a desplegar el menú principal** - **Hacer clic en la pestaña de Gmail**
Criterios de aceptación: El usuario deber poder inciar sesión con su cuenta de usuario en la aplicación de Gmail

Identificador: U002
Nombre: Acceso HOLDED
Tipo: Tipo de prueba, unitaria
Descripción: El usuario entra en la aplicación Holded de facturacion donde puede ver todas sus facturas, clientes y proyectos para acceder a ellos.
Pre-condición: El usuario debe tener disponible el icono de la aplicación para abrirla.
Instrucciones de prueba: - **Abrir la aplicación mediante su icono en el escritorio** - **Acceder a desplegar el menú principal** - **Hacer clic en la pestaña de Holded**
Criterios de aceptación: El usuario deber poder inciar sesión con su cuenta de usuario en la aplicación de Holded.

Identificador: U003
Nombre: Acceso Salesforce
Tipo: Tipo de prueba, unitaria

Descripción: El usuario entra en la aplicación Salesforce CRM donde puede ver todas sus oportunidades, leads contactos etc…
Pre-condición: El usuario debe tener disponible el icono de la aplicación para abrirla.
Instrucciones de prueba: - **Abrir la aplicación mediante su icono en el escritorio** - **Acceder a desplegar el menú principal** - **Hacer clic en la pestaña de Salesforce**
Criterios de aceptación: El usuario deber poder inciar sesión con su cuenta de usuario en la aplicación de Salesforce.

Dado a que todas las pruebas unitarias siguen el mismo formato procedemos a las pruebas de integración. Todas las pruebas unitarias correspondientes a las plataformas externas no se harán por nuestra parte debito a que confiamos en que el resto de empresas externas las hayan llevado a cabo por su parte.

Pruebas de integración.

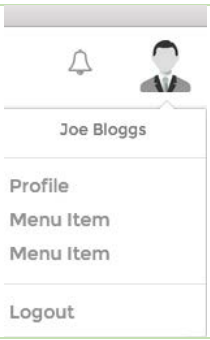
Identificador: I001
Nombre: Acceso correcto a la aplicación.
Tipo: Integración.
Descripción: El usuario entra en la aplicación desarrollada en el proyecto TFG.
Pre-condición: El usuario debe tener disponible el menú de aplicaciones a las que quiere acceder.
Instrucciones de prueba: - **Abrir la aplicación mediante su pestaña única del menú** - **Acceder a desplegar el menú principal** - **Hacer clic en la pestaña correspondiente a la aplicación externa correspondiente.**
Criterios de aceptación: El usuario deber poder iniciar sesión con su cuenta de usuario en cualquier aplicación externa.

Identificador: I002
Nombre: Acceso incorrecto a la aplicación.

Tipo: Integración.
Descripción: El usuario entra en la aplicación desarrollada en el proyecto TFG.
Pre-condición: El usuario debe tener disponible el menú de aplicaciones a las que quiere acceder.
Instrucciones de prueba: - **Abrir la aplicación mediante su pestaña única del menú** - **Acceder a desplegar el menú principal** - **Hacer clic en la pestaña correspondiente a la aplicación externa correspondiente.** - **Incluir el usuario y contraseña incorrectos**
Criterios de aceptación: El usuario deber recibir un mensaje de error

Identificador: I003
Nombre: Salida de la aplicación.
Tipo: Integración.
Descripción: El usuario entra y cierra la aplicación desarrollada en el proyecto TFG.
Pre-condición: El usuario debe tener disponible en la pestaña de la aplicación principal la posibilidad de cerrar el proceso.
Instrucciones de prueba: - **Abrir la aplicación mediante su pestaña única del menú** - **Cerrar la aplicación**
Criterios de aceptación: El usuario deber poder cerrar la aplicación en cualquier tipo de sistema operativo.

Identificador: I004
Nombre: Editar perfil de administrador.
Tipo: Integración.
Descripción: El usuario entra y accede a la administración del perfil de la aplicacion.

Pre-condición: El usuario debe tener disponible en la pestaña de la aplicación principal la posibilidad acceder al perfil de usuario	
Instrucciones de prueba: - **Abrir la aplicación mediante su pestaña única del menú** - **Acceder al perfil de usuario**	
Criterios de aceptación: El usuario deber poder acceder o abrir la aplicación en cualquier tipo de sistema operativo.	

Capitulo 8. Conclusiones.

En este proyecto hemos descrito la forma en que hemos realizado el proyecto desde una fase primera en la que hemos realizado un análisis sobre el diseño de la aplicación, pasando por una fase de desarrollo hasta una fase final de pruebas en la que verificábamos que realmente el software que se estaba construyendo era usable y cumplía unas métricas de calidad.

No solo se ha llevado un profundo trabajo de investigación sobre diferentes plataformas de software empresarial, sino que se considera que hemos unido las mejores de ellas según varios requisitos y las hemos plasmado en una única aplicación de escritorio.

Gracias a nuestra aplicación las empresas podrían eliminar de sus ordenadores los navegadores tradicionales, haciendo más productiva la actividad del día a día de sus empleados.

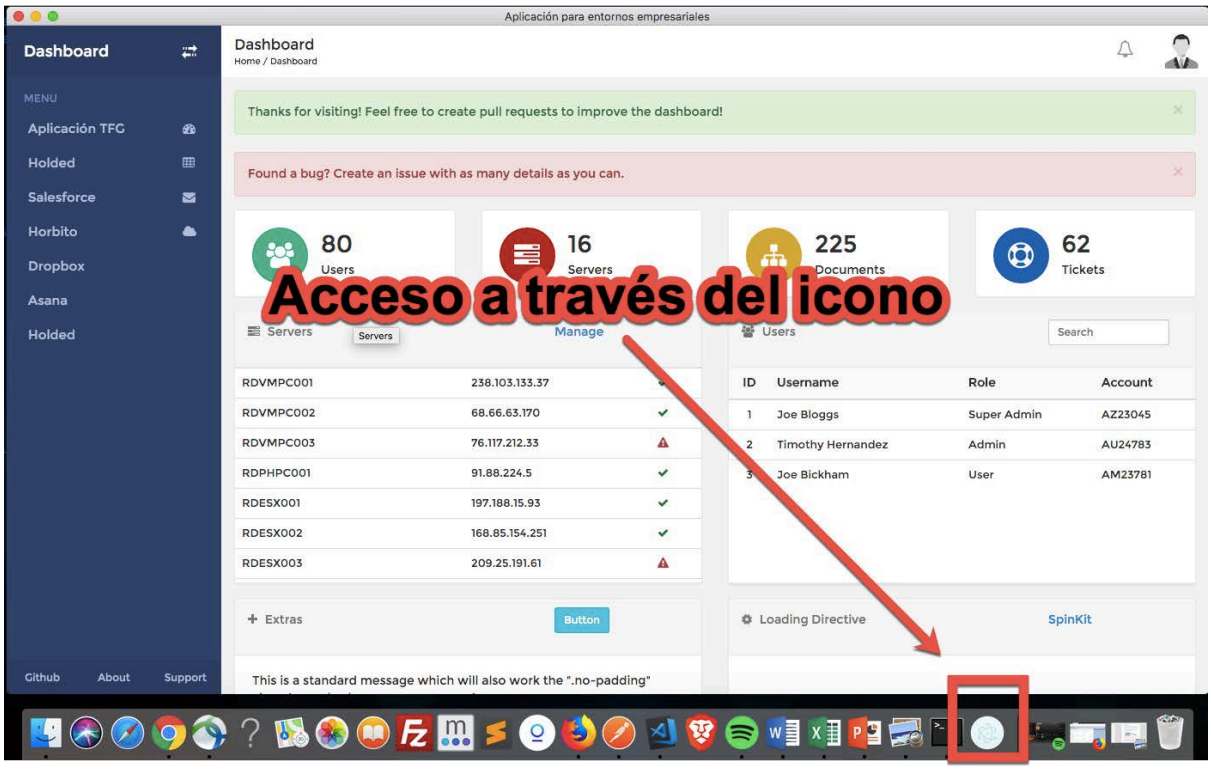
Tras este documento y el con el previo desarrollo de la aplicación podemos afirmar dos cosas, la primera de ellas que por un lado hemos conseguido desarrollar un producto útil y siguiendo los estándares de calidad tanto en diseño como en arquitectura. Y por otro lado hemos llevado a cabo un proyecto cumpliendo con todas las fases necesarias.

La aplicación final ha resultado ser desarrollada con bastante claridad e muy intuitiva para el usuario final.

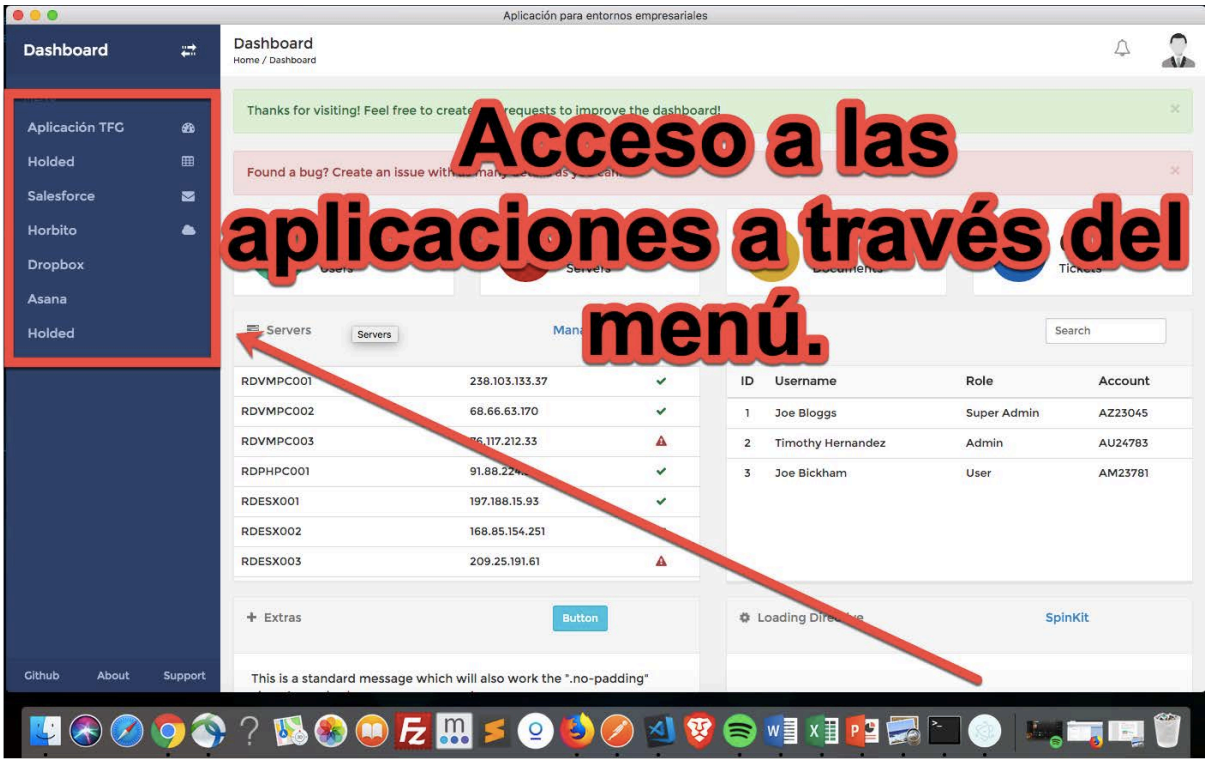
A pesar de ello se pretende llevar a cabo una segunda iteración en la que se puedan añadir a la aplicación diversas funcionalidades.

Capitulo 8. Manual de usuario de la aplicación.

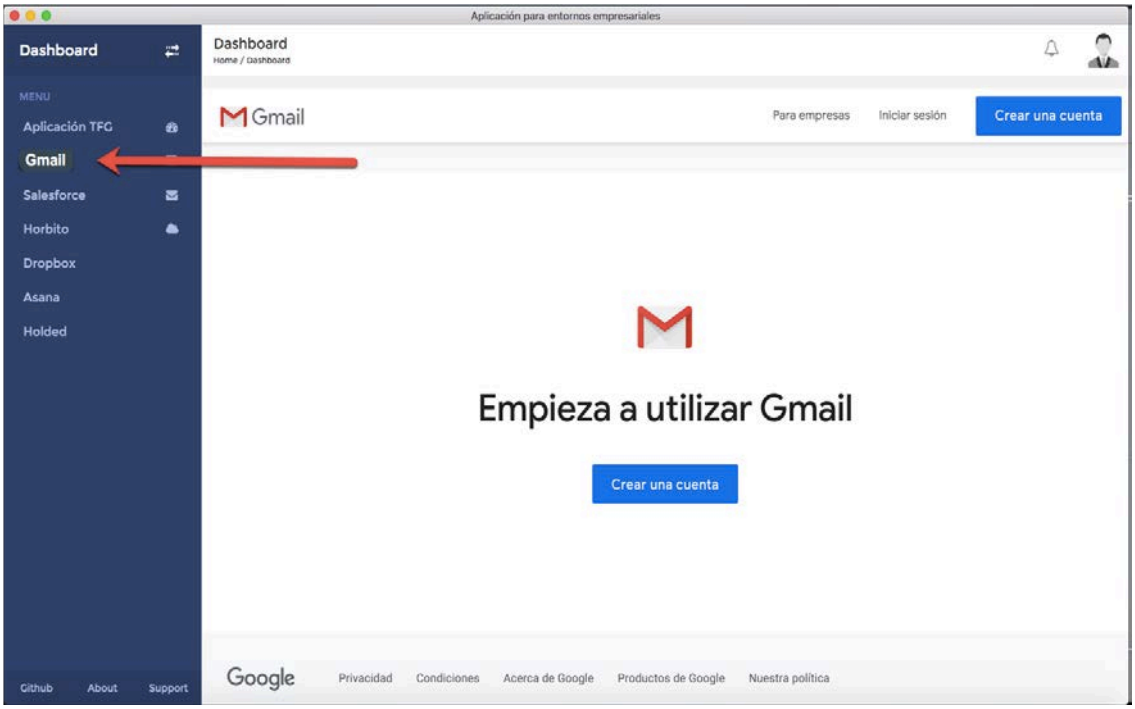
1. Accedemos a la aplicación mediante el icono que tenemos en el escritorio



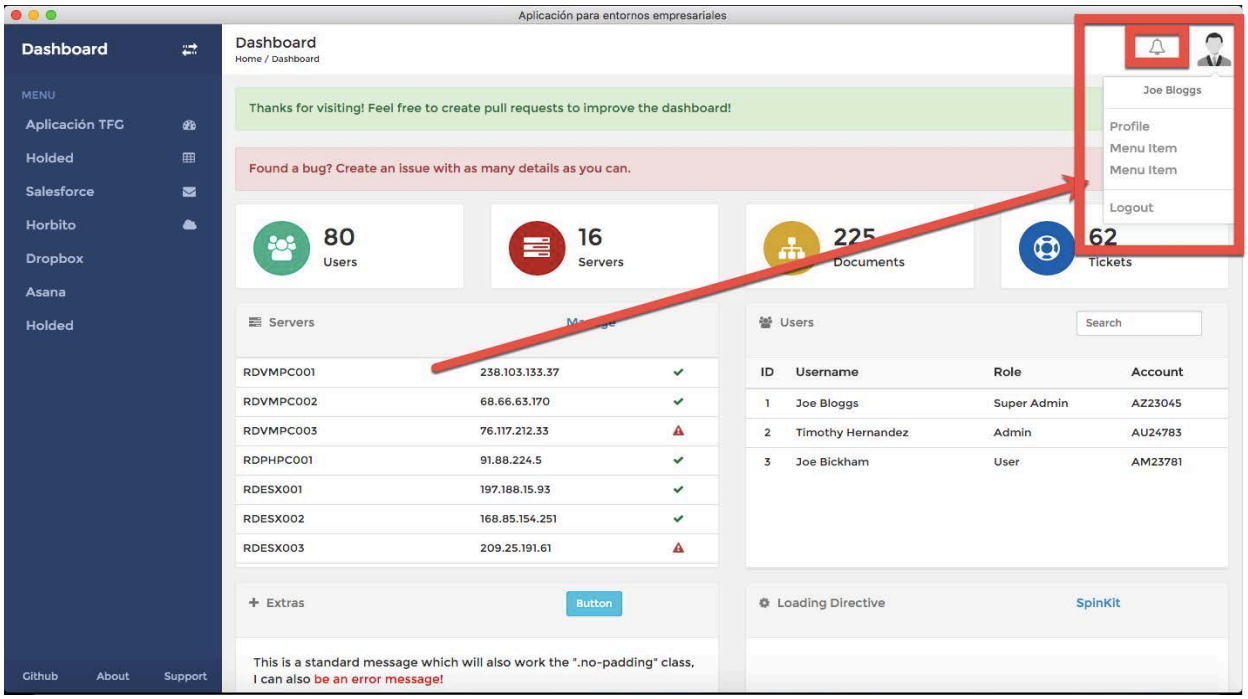
2. Acceso a las diferentes plataformas externas gracias al menú principal.



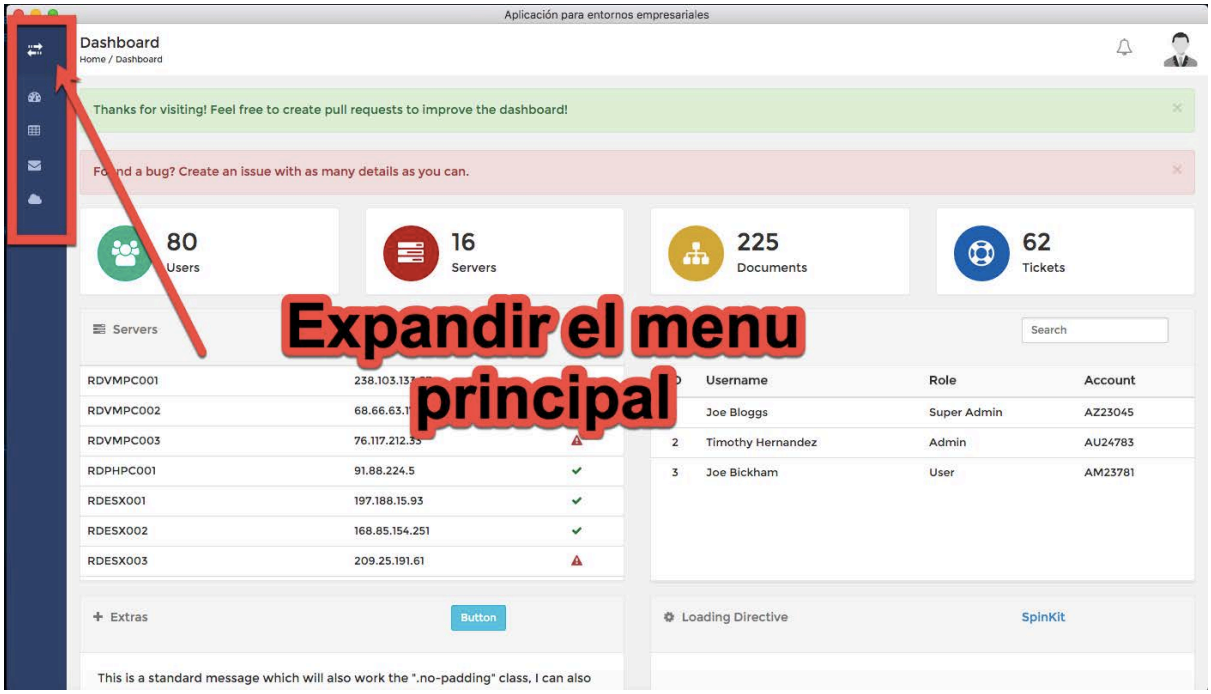
3. Acceso a la plataforma de Gmail gracias al menú de aplicaciones.



4. Acceso al cuadro de mandos de la aplicación, notificaciones y perfil de usuario.



5. Expandir y esconder el menú principal.



Tal y como podemos ver no es necesario incluir mucha información sobre el funcionamiento de la aplicación. Por un lado, hemos conseguido unificar el acceso a las aplicaciones a raíz de un solo menú. Por otro lado, el funcionamiento de cada plataforma que se ha añadido es completamente intuitiva y fácil de usar.

A continuación, incluimos los enlaces o lugares donde se puede encontrar información para navegar sobre estas plataformas:

Gmail	https://support.google.com/mail/forum/AAAAK7un8RUqmO0CzR90OE/?hl=en&gpf=d/topic/gmail/qmO0CzR90OE
Holded	https://www.holded.com/es/como-empezar
Salesforce	https://help.salesforce.com/apex/HTViewHelpDoc?id=getstart_help.htm&language=en_us
Horbito	https://elandroidelibre.elespanol.com/2017/03/horbito-te-da-ordenador-la-nube-25gb-gratis.html
Dropbox	https://www.dropbox.com/es/guide/business
Telegram	https://telegram.org/faq/es

Capitulo 9. Referencias.

1. https://www.elmundo.es/economia/2016/05/23/573e0e7846163f557a8b45eb.html
2. https://help.salesforce.com/apex/HTViewHelpDoc?id=getstart_help.htm&language=en_us
3. https://blog.prototypr.io/functional-and-nonfunctional-requirements-specification-and-types-440a63a8dfdb
4. https://www.i-scoop.eu/digital-transformation/
5. https://uxdesign.cc/the-user-experience-of-choosing-the-simplest-possible-words-90628a3c4a44
6. https://techcrunch.com/2019/04/24/holded/
7. https://en.wikipedia.org/wiki/Electron
8. https://www.tandfonline.com/doi/full/10.1080/09537325.2017.1390220
9. https://www.webopedia.com/TERM/C/cloud_computing.html
10. https://www.programmableweb.com/api/telegram
11. https://www.rubydoc.info/github/gmailgem/gmail
12. https://www.iebschool.com/blog/que-es-transformacion-digital-business/
13. https://www.powerdata.es/transformacion-digital
14. https://www.cegosonlineuniversity.com/la-transformacion-digital-en-el-mundo-empresarial/
15. https://www.marcvidal.net/transformacion-digital

Capitulo 10. Declaración de originalidad.

Yo, **Pablo Camprubi Garcia** declaro que el Trabajo de fin de Grado presentado en la Universidad Carlos III de Madrid "**Aplicación para entornos empresariales**" es original mío, que no ha sido presentado en ninguna otra universidad como TFG y que todas las fuentes que han sido utilizadas han sido publicadas adecuadamente y se han citado como merecen.

Colmenarejo, a 17 de junio de 2019

Firma:

Alumno de Doble Grado en ADE e Ingeniería Informática